SNNAP

(SIMULATOR FOR NEURAL NETWORKS AND ACTION POTENTIALS)

Tutorial

January 2003

The University of Texas-Houston Medical School
Center for Computational Biomedicine
Department of Neurobiology and Anatomy
Houston, TX 77030

http://snnap.uth.tmc.edu

© 1993-2003 The University of Texas Health Science Center at Houston

Table of Contents

Chapter 1: Introduction to *SNNAP* 5
Introduction to Tutorial 6
Outline 6
Typographic Conventions 7
Overview of the Capabilities of SNNAP 8
Some Example Simulations to Illustrate the Capabilities of SNNAP 9
SNNAP Parameters and Units 13
Chapter 2: Getting Started 14
Instructions for Installing SNNAP 15
Structure of Tutorial Examples Directory 19
Chapter 3: The Hodgkin-Huxley Neuron Model 21
Hodgkin-Huxley Model 21
Running the Hodgkin-Huxley Model With SNNAP 23
Launching SNNAP 23
Running a Simulation 24
Printing Simulation Results 27
Changing Simulation Parameters 28
Adding Current Injection 28
Changing Model Parameters 30
Changing Model Conductance Parameter 31
Changing Simulation Duration 34
Changing Output Display 35
Chapter 4: Bursting Neurons and Central Pattern Generators 38
Introduction to Bursting Neurons 39
The Morris-Lecar Spiking Neuron Model 39
Simulation of the Morris-Lecar Model 40
Description of the Morris-Lecar Model 41
Adding Ionic Currents 42
Ion Pools 46
Setting Ion Equation 52
Changing Output Screen 57
Adding New Output Variable 58
Multiple Output Graphs 59
Displaying Ion Currents 62

Central Pattern Generator Modulation 65
Chapter 5 Modeling Small Neuronal Networks 67
Introduction to Small Neuronal Networks 68
Running a Small Neuronal Network Simulation 68
Synaptic Connections 70
Changing Synaptic Strength 75
Building Networks 78
Adding a Neuron to a Network 79
Adding a Synaptic Connection in a Network 81
Conclusions 85
Appendix: References 86
Published Studies That Used SNNAP 86
Literature Cited in Manual 88

Disclaimer

SNNAP is distributed **as is**. This program is distributed in the hope that it will be useful, but **without any warranty**; without even the implied warranty of **merchantability** or **fitness** for a particular purpose. The authors make no claims as to the performance of the program. Individuals who wish access to the source code should contact Douglas.Baxter@uth.tmc.eud.

CHAPTER 1: INTRODUCTION TO *SNNAP*

Recently, there has been a dramatic increase in the number of neurobiologists using computational methods as an adjunct to their empirical studies. As experimental data continue to amass, it is increasingly clear that physiological and anatomical data alone are not enough to infer how neural circuits work. Researchers are recognizing the need for a quantitative modeling approach to explore the functional consequences of neuronal and network features. Computer simulations are an increasingly important tool for neuroscience research.

SNNAP (Simulator for Neural Network and Action Potentials) was designed as a tool for the rapid development and simulation of realistic models of single neurons and small neural networks. With SNNAP, all aspects of developing and running simulations are controlled through a user-friendly graphical interface. Thus, no programming skills are necessary to develop and run simulations. The electrical properties of individual neurons are described with either Hodgkin-Huxley type voltage- and time-dependent ionic currents or integrate and fire models. The connections among neurons can be made by either an electrical, modulatory or chemical synapse. The chemical synaptic connections are capable of expressing many forms of plasticity, such as homo- and heterosynaptic depression and facilitation. SNNAP also includes descriptions of intracellular second messengers and ions, which in turn, can be linked to ionic conductances or mechanisms regulating synaptic transmission. Thus, you can use SNNAP to simulate the modulation of cellular and synaptic properties. SNNAP can also be used to simulate the flow of current in multi-compartment models of cell. Many common experimental manipulations can be simulated, such as injecting current, voltage clamping and applying modulatory transmitters. SNNAP (since version 5) was implemented in the Java programming language. Thus, SNNAP can run on virtually any computer and with most operating systems.

INTRODUCTION TO TUTORIAL

This tutorial was designed for non programmers who are interested in carrying out computer simulations of neuroscience experiments. This hand-on tutorial will guide you through several examples using SNNAP. Each chapter was designed to introduce you to several basic SNNAP capabilities in a particular domain of neuroscience. You will learn how to use SNNAP to write mathematical models and run computer simulations in the various domains described in each chapter. This tutorial may complement an introduction to computational neuroscience or be a basis for further investigation.

Outline

This manual is a hands-on tutorial, designed to provide you with experience in using SNNAP as well as introduce you to several topics in computational neuroscience.

- Chapter 1: *Introduction* presents an overview of SNNAP and this tutorial as well as a short description of SNNAP highlights.
- Chapter 2: *Getting Started* describes how to download and install the SNNAP software.
- Chapter 3: *Hodgkin-Huxley Neuron Model* presents the classic neuronal model of Hodgkin and Huxley and shows how to run a simulation using SNNAP. This chapter also describes how to change basic simulation parameters and model parameters.
- Chapter 4: *Bursting Neurons and Central Pattern Generators* describes a more complex model than the Hodgkin-Huxley model, while introducing the concept of an ion pool and second messenger, and how such elements can be used to modulate neuronal behavior with SNNAP.
- Chapter 5: *Small Networks* presents a three neuron network of Hodgkin-Huxley models and shows how to build a network with synaptic connections using SNNAP. The network model can display oscillatory behavior because of the network architecture and synaptic connectivity.

Typographic Conventions

The following typographic conventions apply throughout this manual:

- Code extracts and file names are written in `this typeface.`
- Italic type is used to indicate user-specific information. Also, important ideas are emphasized like *this*.
- Values that you must fill in (for example, a file name or a path name) also appears in the same typeface as code extracts but slanted to indicate you must supply an appropriate value; for example, *SNNAPhome* indicates that you must fill in a value for `SNNAPhome`.
- Square brackets ([]) indicate optional items.
- Ellipsis (...) indicate that you can repeat the information.
- A vertical bar (|) indicates a choice within braces ({}) or brackets ([]).

The following vocabulary is used in this manual to explain how to navigate the menus of SNNAP:

- Right-click: click the right mouse button.
- Left-click: click the left mouse button.
- Click: click either the right or left mouse button.
- *Menu*>*SubMenu*$_0$>…>*SubMenu*$_n$>*Item*: menu item identified by the path to access it.
- ***SNNAPhome*** is the location where you have installed the SNNAP software. All locations presented in the manual start from *SNNAPhome*.

OVERVIEW OF THE CAPABILITIES OF SNNAP

A brief list of some of the capabilities of SNNAP (version 8) is provided below:

- ✓ SNNAP can simulate networks of up to 10000 neurons with electrical, chemical and modulatory synaptic connections.
- ✓ SNNAP can simulation networks containing both Hodgkin-Huxley type neurons and Integrate-and-Fire type cells. Moreover, the 'synaptic' contacts among integrate-and-fire cells can incorporate 'learning rules' that modify the synaptic weights. The User is provided with a selection of several non associative and associative learning rules.
- ✓ SNNAP can simulate the flow of current in multi-compartment models of neurons, which in turn, can be incorporated into neural networks.
- ✓ SNNAP can simulate intracellular pools of ions and / or second messengers that can modulate neuronal processes such as membrane conductances and transmitter release. Moreover, the descriptions of the ion pools and second-messenger pools can include serial interactions as well as converging and diverging interactions. For example, the synthesis of a second messenger (e.g., cAMP) can be regulated by both a modulatory transmitter and the levels of intracellular Ca^{2+}.
- ✓ The number of ionic conductances as well as the number of intracellular pools of ions and second messengers that can be incorporated into a neuron is dynamically allocated. The User can add as many elements to a model as may be necessary to describe a given neuron. Thus, models of neurons can achieve a high level of sophistication.
- ✓ Chemical synaptic connections have User-defined kinetics (i.e., fast or slow), can produce either increases or decreases in postsynaptic conductance, can be excitatory or inhibitory, can have multiple components (e.g., fast and slow components, increase and decrease conductance components, excitatory and inhibitory components, etc.), and can manifest homosynaptic plasticity (i.e., depression, facilitation, or both).
- ✓ Chemical synaptic connections can include a description of a pool of transmitter that is regulated by depletion and / or mobilization and that can be modulated by intracellular concentrations of ions and second messengers. Thus, the User can simulation heterosynaptic plasticity.
- ✓ Descriptions of synapses (both chemical and modulatory) can include a voltage-dependent component. For example, a synapse can include a NMDA-like conductance.
- ✓ SNNAP can simulate asymmetrical electrical coupling between cells.
- ✓ SNNAP can simulate a number of experimental manipulations, such as injecting current into neurons, voltage clamping neurons, and applying modulators to neurons. In addition, SNNAP can simulate noise within any conductance (i.e., membrane, synaptic, or coupling conductances) and SNNAP can simulate a novel procedure for 'clamping a state variable' in which the magnitude of a specified membrane current(s) can be clamped to a specified value at any given time.
- ✓ SNNAP includes a `Batch Mode` of operation, which allows the User to assign any series of values to any given parameter or combination of parameters. The `Batch Mode` automatically reruns the simulation with each new value and displays, prints, and / or saves the results.
- ✓ The on-screen display can plot any combination of state variables in either the time domain and / or as a phase plane (i.e., one state variable vs. another). Moreover, the results of a simulation can be printed,

stored as a postscript file or stored as data file. The data file is in ASCII format and can be used by other software packages and / or by the `Off-Line Viewer` that is provided with SNNAP.

- ✓ SNNAP includes a large suite of example simulations that illustrate capabilities of SNNAP and that can be used as a tutorial for learning how to use SNNAP or as an aid for teaching neuroscience.
- ✓ In addition to the example simulations, other supplemental information and software that is included with SNNAP including an electronic version (i.e., *.pdf) of the User's Manual, numerous Excel® spreadsheets, illustrates of simulations, and new program called CellMatrix. CellMatrix is a data-management tool for organizing empirical data that describe synaptic connections among identified neurons. This tool can be useful in developing models of small neural networks that are based on a large body of published literature.
- ✓ SNNAP was implemented in Java and can run on virtually any computer and under most operating systems.
- ✓ SNNAP is freely available and can be downloaded via the internet. The software, example files and User's manual are available at **http://SNNAP.uth.tmc.edu**.

SOME EXAMPLE SIMULATIONS TO ILLUSTRATE THE CAPABILITIES OF SNNAP

Detailed descriptions of the capabilities of SNNAP are provided in **Chapter 3** and **Appendix B**, which describe the equations incorporated into SNNAP and the example simulations that are distributed with SNNAP. To briefly illustrate some capabilities and potential uses of SNNAP, a few example simulations are presented below. Figure 1 illustrates how SNNAP can be used to simulate the biophysical properties of neurons, including relatively simple models of the action potential and neuronal excitability (e.g., the Hodgkin-Huxley model of the squid giant axon) and more complex models of autonomous bursting, intracellular second messengers, and modulation (e.g., the Butera et al. model of the R15 neuron in *Aplysia*).

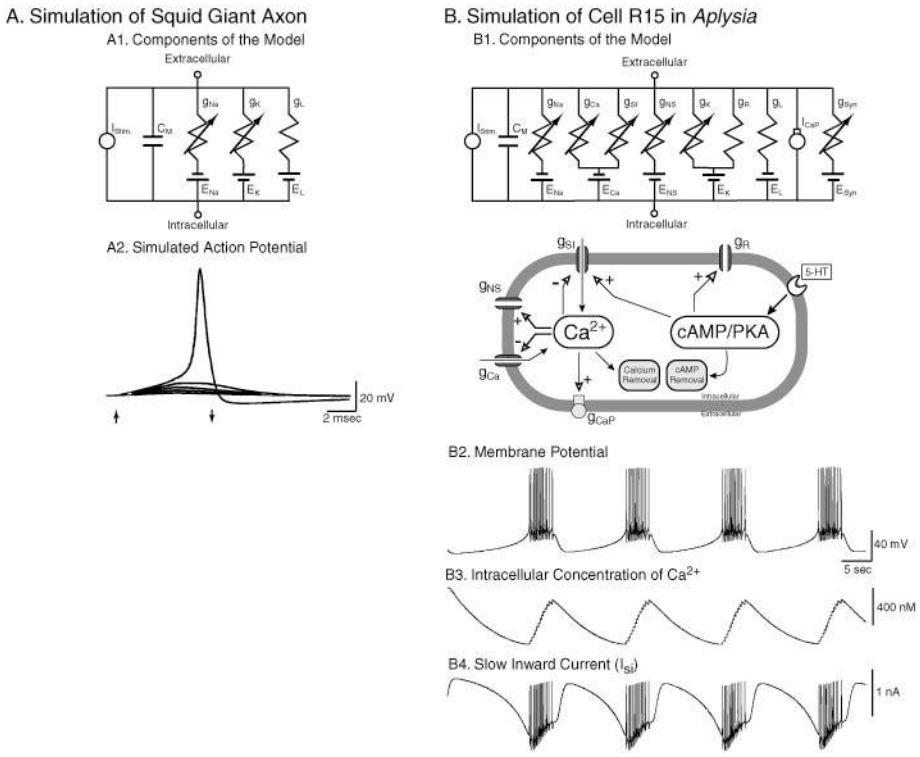

Figure 1.1. Using SNNAP to model neurons with relatively simple properties (e.g., the squid giant axon) or to model neurons with more complex properties such as autonomous bursting, second messengers and modulation (e.g., R15). The equivalent circuit diagrams for both models are illustrated (**A1** and **B1**). In addition, the intracellular regulatory pathways of the R15 model are illustrated (**B1**). **A:** Simulation of the Hodgkin and Huxley model of an action potential in the squid giant axon. This model has only two voltage- and time-dependent conductances. **A2**: The SNNAP simulation illustrates the `Batch Mode` of operation. In the `Batch Mode`, simulations are repeated automatically while systematically varying parameter values. In this example the magnitude of the injected depolarizing current pulse (arrows) was increased with each simulation and the results of each simulation were superimposed. The SNNAP input files used to generate this simulation are included in the `/Examples /H_H_type_neurons /Biophysics_01` subdirectory. **B:** Simulation of the Butera et al. model of the bursting neuron R15 in *Aplysia.* This model incorporates six voltage- and time-dependent conductances. In addition, it incorporates two intracellular pools (i.e., an ion pool of calcium and a pool of the second message cAMP). These pools, in turn, modulate several membrane conductances. **B2:** SNNAP allows the User to plot any variable in a model, such as the membrane voltage, intracellular concentration of calcium and specific membrane currents. The SNNAP input files used to generate this simulation are included in the `/Examples /H_H_type_neurons /R15` subdirectory.

In addition to simulating the complex biophysical properties of neurons (Figure 1), SNNAP can simulate the complexities of synaptic connections and synaptic plasticity (both homo- and heterosynaptic plasticity). Several examples are illustrated in Figure 1.2. SNNAP can simulate synaptic connections with homosynaptic facilitation and / or depression, synaptic connections with multiple components (e.g., fast and slow PSPs),

synaptic connections that produce decreased postsynaptic conductances as well as synaptic connections that are modulated via heterosynaptic connections. In addition, SNNAP can simulate modulatory synaptic connections (i.e., synaptic connections that drive the synthesis of second messengers in the postsynaptic neuron, not shown) and synaptic connections that are both voltage- and time-dependent (not shown). For example, SNNAP can simulate NMDA-type synaptic responses.

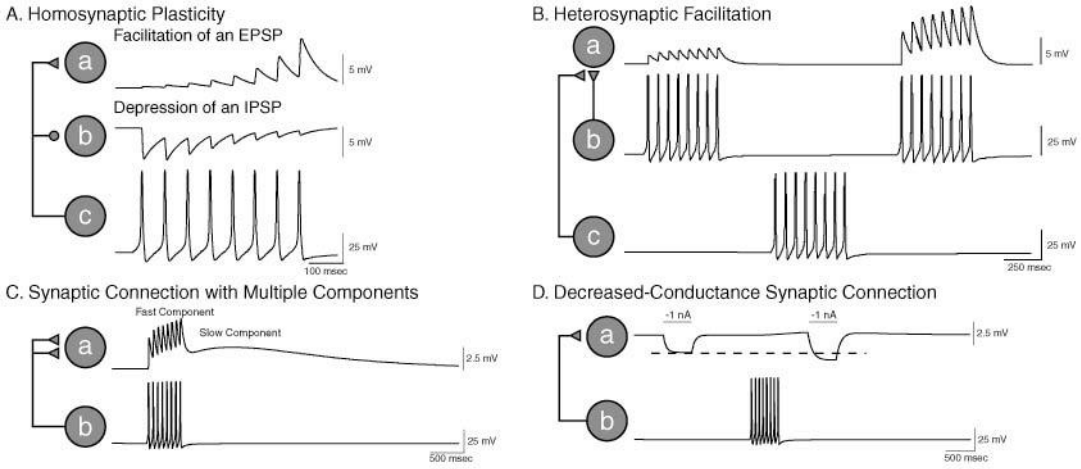

Figure 1.2. Simulating synaptic connections and synaptic plasticity with SNNAP. Many different types of synapses and plasticity can be modeled. For example, homosynaptic facilitation or depression (**A**) can be simulated. By including a second messenger system that modulates transmitter release, heterosynaptic plasticity (**B**) can be simulated. SNNAP can simulate synaptic responses that have multiple components (**C**) such as fast and slow potentials, and synaptic response that induce conductance decreases (**D**). The `/Examples /Synaptic_connections` subdirectory contains simulations similar to those illustrated in this figure.

Although SNNAP was designed to simulate neurons as a single, isopotential compartment, it can also simulate neurons as multi compartmental structures (Figure 1. 3). The fundamental computational unit of a SNNAP simulation is the *.neu file (i.e., neuron file). The *.neu files can be used to represent a single neuron or to represent individual compartments of a multi-compartment neuron. To develop a multi compartment model, the properties of the *.neu files are adjusted to match the morphological features of the neuronal compartments. These *.neu files are then linked together so as to reflect the branching structure of a given neuron. SNNAP incorporates several tools that help the User develop multi compartmental models. For example, the User can enter geometric parameters (e.g., the diameter and length of

compartment) and SNNAP calculates the membrane area and appropriately scales the ionic conductances for each compartment.

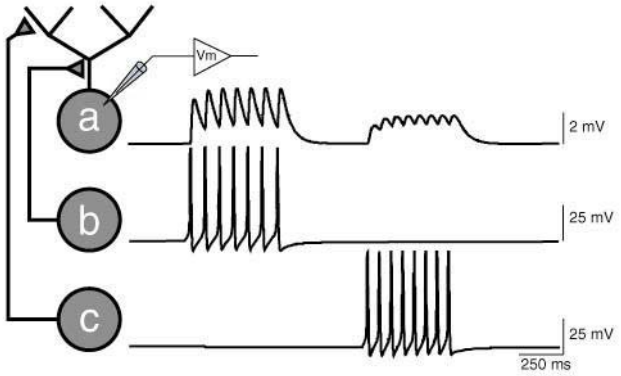

Figure 1. 3. Simulating a neuron with multiple compartments**.** SNNAP can be used to model the spread of current in a multi-compartment model of a neuron. In this example, the postsynaptic neuron (neuron A) was modeled as a branching structure with 11 compartments of progressively decreasing size (i.e., diameter and length). The properties of the two presynaptic inputs (i.e., neurons B and C) were identical. One input, however, was further from the soma than the other. The voltage of the postsynaptic neuron was monitored in the soma compartment (Vm) and the two presynaptic neurons were stimulated. The EPSPs for the more distal synaptic input (i.e., from neuron C) were attenuated and their kinetics appeared slower. The `/Examples /Compartmental_models` subdirectory contains simulations similar to those illustrated in this figure.

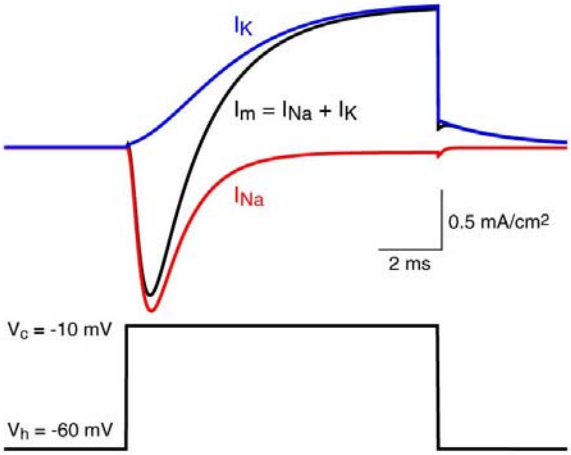

Figure 1.4. SNNAP can simulate many common experimental manipulation, such as voltage-clamping. In this example, the Hodgkin-Huxley model of the squid giant axon as held at –60 mV and stepped to –10 mV. The total membrane current (black trace), sodium current (red trace) and potassium current (blue trace) that were elicited during the step are illustrated. The files necessary to run a simulation such as this are in the /Examples /HH_type_neuron /Biophysics_01 subdirectory

As illustrated in Figure 1.4, SNNAP also allows the User to simulate many common experimental manipulations, such as injecting current into neurons, voltage-clamping and applying modulatory transmitters. SNNAP also offers several rather unique manipulations, including "clamping" a given membrane current to a specified value, injecting complex signals into neurons (e.g., sine waves, ramps, exponential waveforms) and introducing noise into membrane and / or synaptic conductances. The User sets the magnitude and frequency of the stochastic fluctuations and can specific whether there is a single source of noise (i.e., all fluctuations occur in unison) or there are multiple and independent sources of noise.

SNNAP PARAMETERS AND UNITS

The following table presents the parameters and units used by the SNNAP simulator.

Parameter	**Unit**
Time	Seconds
Conductance	microSiemens (uS)
Current	microAmps (uA)
Capacitance	microFarads (uF)
Cell diameter	microMeter (uM)
Membrane Potential	milliVolts (mV)

CHAPTER 2: GETTING STARTED

In order to run the SNNAP application you must have a Java virtual machine installed on your computer. The SNNAP software is provided as a Java jar file. You can therefore place it in any folder or directory you wish and easily launch the application.

The SNNAP software provides you with a tutorial that covers several computational neuroscience examples. The tutorial manual and examples are provided in the folder `tutorial`.

This chapter describes how to install SNNAP, as well as a Java environment, if you do not already have one. The directory structure for the tutorial examples are also described.

This chapter consists of the following sections:

- *Instructions for Installing SNNAP* describes how to install the software on machines running Windows, MacOS or Linux.
- *Tutorial Examples Directory Structure* presents the example files of the tutorial and their structure.

INSTRUCTIONS FOR INSTALLING SNNAP

Hardware and Software Requirements. About 25 megabytes of hard disk space are needed to accommodate SNNAP and its associated files. These additional files include items such as this manual, and extensive set of Example simulations, which can be used as an informal tutorial on how to use SNNAP and as a starting point for developing new simulations, an additional program termed CellMatrix, which can be used to manage empirical data, and several Excel® spread sheets, README files and images of simulations. In addition, SNNAP requires a functioning version of Java on the user's computer. This may be either the Java Runtime Environment (JRE) or the Java Development Kit (JDK). It is recommended that the user install the most current version of Java and that it be installed such that Java programs can be run from any directory.

Structure of SNNAP. The `SNNAP8` directory contains two Java `jar` files (`SNNAP8.jar` and `CellMatrix.jar`), a icon file (`SNNAP8.ico`), which can be used as a desktop icon, and two subdirectories (`examples` and `tutorial`), which both contain many additional subdirectories. The `SNNAP8.jar` contains all files necessary to run SNNAP.

Installing SNNAP. SNNAP is distributed as a `jar` file. The `SNNAP8.jar` file contains all of the `*.class` files necessary to run SNNAP. The `*.class` files contain the *byte code* used by the JRE on any given computer. Briefly, Java applications are created and run by: first writing 'source code' just as in any programming language. Second, the JDK (Java Development Kit) is used to compile this source code into byte code. Byte code is not a machine-specific binary file, however, as with standard compilers. Rather byte code is an intermediate or 'generic' form of the program that is used by the JRE (Java Runtime Environment). The JRE are specific to each type of computer and operating environment. The user must locate and install the appropriate JRE for their computer system. The JRE provides a 'virtual machine' within which the byte is run. Finally, the user invokes the JRE and runs the Java application, such as SNNAP. Thus, the user installs the JRE that is specific to their computer and runs a 'generic' version of SNNAP byte code. Java programs are 'machine and operating

system independent' to the extent that Java is supported for each type of computer and operating environment.

Because SNNAP is distributed as a `jar` file, no installation *pre se* is necessary. You can simply copy the `SNNAP8.jar` file and its associated subdirectories onto your computer. It is necessary, however, to have Java installed (see below). Java and the necessary installation instructions can be obtained free-of-charge from the Sun web site (http://java.sun.com). To test whether the Java installation was successful, use the command-line interface of your operating system and simply types 'java -version' at the command-line prompt. If Java is installed properly, the window will display the version of the installed Java environment, similar to Fig. 2.1. Once Java is installed and SNNAP has been copied onto your computer, running SNNAP can be as simple as double clicking on the `SNNAP8.jar` file.

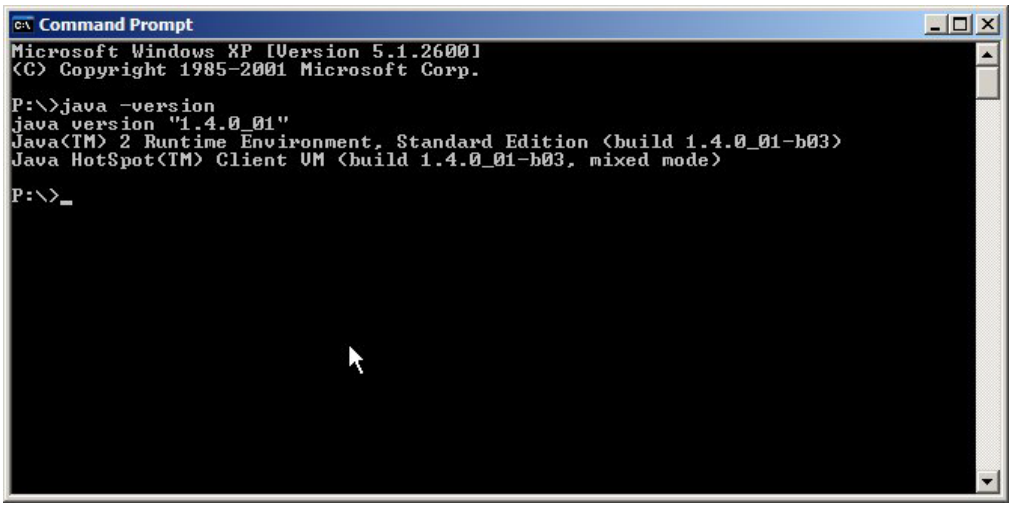
Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
P:\>java -version
java version "1.4.0_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.0_01-b03)
Java HotSpot(TM) Client VM (build 1.4.0_01-b03, mixed mode)
P:\>

Figure 2-1 Testing the Installation of Java. To test whether Java is properly installed, the User must first access the command-line interface of their operating system. All Java applications are initiated from the command-line interface. In Microsoft® Windows, the command-line interface is often referred to as a 'DOS' window. In UNIX environments, the command-line interface can be accessed via a 'console' or 'terminal' window. To test whether Java is properly installed, the User simply types 'java -version' at the command prompt. If Java is installed properly, there should be a response similar to that illustrate above. Otherwise, the operating system will respond with an error message. If the response is an error message, the User should refer to the installation instructions provide with Java. It may be necessary to modify some environment variables of the operating system (e.g., the PATH or User PROFILE).

Installing SNNAP for specific operating environments. The specific procedures for installing Java and SNNAP are listed for three operating systems: Microsoft®

Windows, MacOS® and Linux/UNIX. These instruction can also be found on the SNNAP web site (i.e., http://snnap.uth.tmc.edu).

1) Downloading and installing Java and SNNAP on computers with the **Microsoft®** operating system.

i) Download and install latest version of Java (Java is necessary to run SNNAP)

a. Go to the website http://java.sun.com
b. Select: *Standard Edition J2SE*
c. Select: *J2SE Downloads*
e. Select: *J2SE 1.4.1* (or latest version)
f. Select from table: *Windows SDK Downloads*
g. Save Java 2 on the directory of your choice (for example: `C:\program files\JDK2`)
h. Open the directory with the Java download, find and click the Java JDK2 icon to install Java

ii) Download WinZip (A 'zip' program is necessary to 'uncompress' SNNAP. You may already have a zip program and can skip this step.)

a. Go to the website http://www.winzip.com
b. Download and install WinZip.

iii) Download SNNAP

a. Open the website http://snnap.uth.tmc.edu
b. Click download and click the version you desire
c. Save SNNAP in a folder of your choice, but the directory path may NOT contain a blank character (e.g. `\Program Files` is NOT a valid folder).
d. Find the downloaded SNNAP file in the directory
e. Click the SNNAP icon
f. Save the unzipped SNNAP files in a directory (e.g. C:\)

iv) Run SNNAP on Windows 95/98/NT/2000/XP

There are two methods for initiating a SNNAP simulation session. The first is the simplest. Using your file explorer, go to the folder where SNNAP is located, double click on the `SNNAP8.jar` file and the main control panel for SNNAP will appear.

Alternatively, open a command-line window (see Fig. 2.1), change directory to where SNNAP is located and type:

```
C:\SNNAP8>java -jar snnap8.jar
```

and the main control panel for SNNAP will appear. By using this second method of running SNNAP, if there is a problem with a simulation, error messages will be displayed in the command-line window.

v) Make a shortcut for SNNAP

a. Go to SNNAP folder
b. Highlight the SNNAP8.jar file icon
c. While pressing the right mouse button, drag the `SNNAP8.jar` file to the desktop screen and 'drop it'. Select the 'Create Shortcut Here' option and the shortcut for SNNAP is now created.
d. You may wish to alter the appearance of the shortcut on the desktop. To change the icon associated with SNNAP, highlight the SNNAP desktop icon and press the right mouse button.

Select the 'Properties' option, which will invoke a pop-up window. Select 'Change Icon' and select the 'Browse' option. This will allow you to search for and select a new image for the SNNAP desktop icon. An `ico` file is provide with SNNAP and you may use this icon.

2) Downloading and installing Java and SNNAP on computers with the **MacOS X** operating system.

i) Download SNNAP

a. Click Version 8 (or 7) from URL http://snnap.uth.tmc.edu
b. Save SNNAP in a folder of your choice, but the directory path may NOT contain a blank character (e.g. `\SNNAP Files` is NOT a valid folder).
c. Unzip it with Aladdin StuffIt Expander (freeware) program

ii) Install Java SDK for Mac

a. MacOS X bundles Sun's Java 2 Standard Edition (J2SE) version 1.3 or higher.

3. Downloading and installing Java and SNNAP on computers with the **Linux/UNIX** operating system.

i) Download Java 2 to the computer (or the most recent version)

a. Go to website http://java.sun.com
b. Select: *Standard Edition J2SE*
c. Select: *J2SE Downloads*
e. Select: *J2SE 1.4.1* (or latest version)
f. Select from table: *Linux RPM …* or simply *Linux self-extracting files - Downloads*
g. Follow the instructions and save *Java 2 SDK* on the directory of your choice (for example: /tmp)
h. The file has a name like *j2sdk-1_4_1_01-linux-*.bin*. Change to the directory you saved downloaded file (e.g. with *cd /tmp*), do *chmod 774 j2sdk-1_4_1_01-linux-*.bin* to change the permission, then do *./j2sdk-1_4_1_01-linux-*.bin*
i. For RPM format, you will get a file j2sdk-1_4_1_01-linux-*.rpm. Login as root or su to root, then use "rpm -ivh /tmp/*.rpm" to install
j. For GNUZIP tar ball, you will get a directory containing JDK, like *jdk1.4.1_01*. Login as root or su to root, and do *mv jdk1.4.1_01 /usr/local*
h. Edit */etc/profile* and add */usr/local/jdk1.4. 1_01/* at the end of the PATH line (e.g. PATH="/usr/bin:/bin:/usr/X11R6/bin:/usr/local/bin/:/usr/local/java/jdk1.4.1_01/bin/:)

Detailed instructions can be found at http://java.sun.com/j2se/1.4 /install.html

ii) Download SNNAP

a. Open the website http://snnap.uth.tmc.edu
b. Click download and click the version you desired
c. Save SNNAP in a directory of your choice (e.g. /tmp)
d. Go to the directory and issue *unzip SNNAP8.zip* (for Version 8), it will create a *SNNAP8* directory
e. Move *SNNAP8* to desired places, e.g. */usr/local/* with: *mv /tmp/SNNAP8 /usr/local*

iii) Run SNNAP in Linux

a. Login as root and edit */etc/profile*, adding a line at the end: *alias snnap="java -jar /usr/local/SNNAP8/SNNAP8.jar*"
b. Login as a normal user and type *snnap*, SNNAP8 will be launched

STRUCTURE OF TUTORIAL EXAMPLES DIRECTORY

The tutorial contains several examples which are provided to you in specific folders.

The directory structure is as follows:

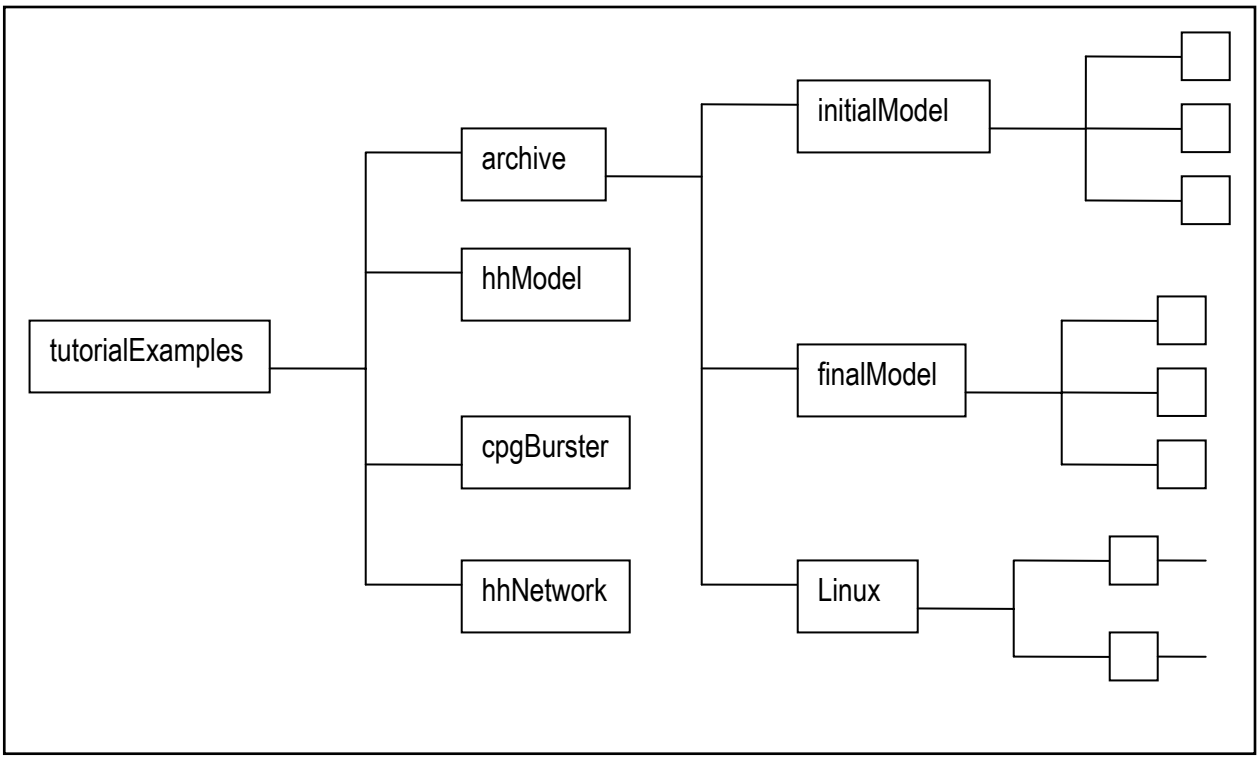

Figure 2.2. Folder structure of tutorial examples.

SNNAP is provided with three tutorial examples. You can find the three examples in folders, `hhModel`, `cpgBurster` and `hhNetwork`. These are the *working folders*, as described in the chapters. The folder `archive` contains copies of the initial model, in folder `initialModel,` as well as the final model, in folder `finalModel,` which corresponds to the final figure shown for each model in the corresponding chapter. Thus you can work on a model in the appropriate working folder and always be able to copy the initial model if needed. Or use the final model to further investigate aspects of SNNAP.

A Linux folder is provided with initial and final folders for `hhNetwork.`The network model uses subfolders which require different pathnames under Windows and Linux.

CHAPTER 3: THE HODGKIN-HUXLEY NEURON MODEL

The Hodgkin and Huxley model was first published in 1952, but still remained the modeling formalism of choice by most neuroscientists. Using the voltage clamp technique that was developed in that day, Hodgkin and Huxley estimated activation and inactivation functions for the sodium and potassium currents. This allowed them to devise a mathematical model which described an action potential that resembled their experimentally recorded action potentials from the squid giant axon.

We will use the Hodgkin and Huxley model to learn how to run a SNNAP simulation, and learn how to alter simulation and model parameters.

HODGKIN-HUXLEY MODEL

The space-clamped version of the Hodgkin-Huxley model consists of four ordinary differential equations. The model describes the change of membrane potential (*V*) with respect to time. Three equations describe the activation (*m* and *n*) and inactivation (*h*) variables of the sodium and potassium currents.

The total membrane current is the sum of a capacity current and an ionic current:

$$I_m = C_m \frac{dV}{dt} + I_i$$

The ionic current is the sum of three individual components: sodium, potassium and leak currents.

$$I_i = I_{Na} + I_K + I_l$$

The sodium current equation is the product of a maximal conductance $\bar{g}_{Na}$, activation variable *m*, inactivation variable *h*, and a driving force $(V - V_{Na})$:

$$I_{Na} = \bar{g}_{Na} m^3 h (V - V_{Na})$$

The potassium current equation is the product of a maximal conductance $\bar{g}_K$, activation variable *n* and a driving force $(V - V_K)$:

$$I_K = \bar{g}_K n(V - V_K)$$

The leak current equation is the product of a maximal conductance $\bar{g}_l$ and a driving force $(V - V_l)$:

$$I_l = \bar{g}_L (V - V_L)$$

The activation variables (*m* and *n*) and inactivation variable *h*, which vary from 0 to 1, follow a standard form, with forward rate functions (α) and backward rate functions (β) and a temperature-dependent factor ϕ:

$$\frac{dm}{dt} = \phi[\alpha_m(V)(1-m) - \beta_m(V)m]$$

$$\frac{dh}{dt} = \phi[\alpha_h(V)(1-h) - \beta_h(V)h]$$

$$\frac{dn}{dt} = \phi[\alpha_n(V)(1-n) - \beta_n(V)n]$$

The forward and backward rate functions, dependent on voltage, where estimated by Hodgkin and Huxley based on their experimental results:

$$\alpha_m(V) = \frac{0.1(-35-V)}{e^{(-35-V)/10} - 1}$$

$$\beta_m(V) = 4e^{(-V-60)/18}$$

$$\alpha_h(V) = 0.07e^{(-V-60)/20}$$

$$\beta_h(V) = \frac{1}{e^{(-30-V)/10} + 1}$$

$$\alpha_n(V) = \frac{0.01(-50 - V)}{e^{(-50-V)/10} - 1}$$

$$\beta_n(V) = 0.125e^{(-V-60)/80}$$

The forward and backward rate equations have been converted from the original HH version to agree with present physiological practice where depolarization of the membrane is taken to be positive. Also, the resting potential has been shifted to -60mV (from the original 0mV).

RUNNING THE HODGKIN-HUXLEY MODEL WITH SNNAP

SNNAP is provided with the Hodgkin-Huxley model in the file: \SNNAPhome\SNNAP8\tutorial \tutorialExamples\hhModel (SNNAPhome is the location where you installed the SNNAP8 directory).

In order to run the model, you will need to launch SNNAP and run the simulation example.

Launching SNNAP

To launch SNNAP, go to the folder SNNAP8. The folder has the following structure:

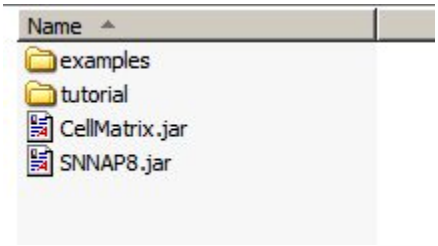

Figure 3.1. The structure of the folder SNNAP8.

Double-click the executable file `SNNAP8.jar` to open the main SNNAP window, see Figure 3.1.

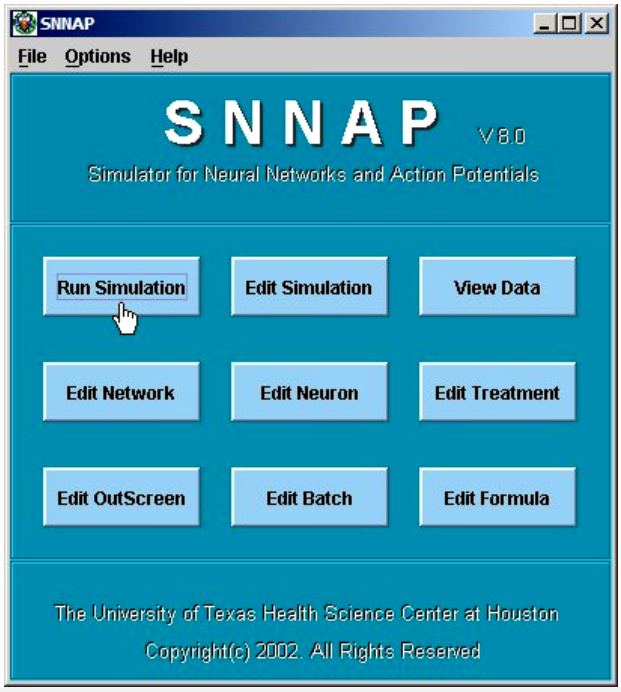

Figure 3.2: Main SNNAP window.

The main SNNAP window provides buttons to open the various functional windows to edit SNNAP parameters, edit model and network parameters and execute simulations.

Running a Simulation

In the main SNNAP window, click the button `Run Simulation` to open the simulation viewing window.

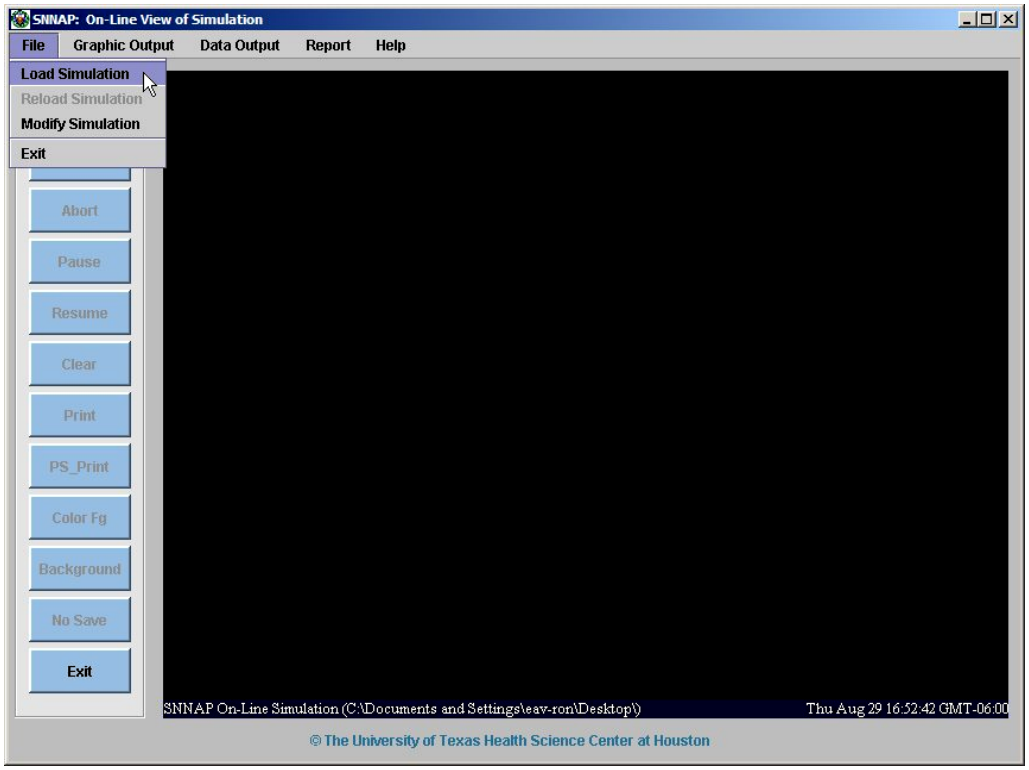

Figure 3.3. Main simulation viewing window.

Click *File>Load Simulation* to open the file chooser window.

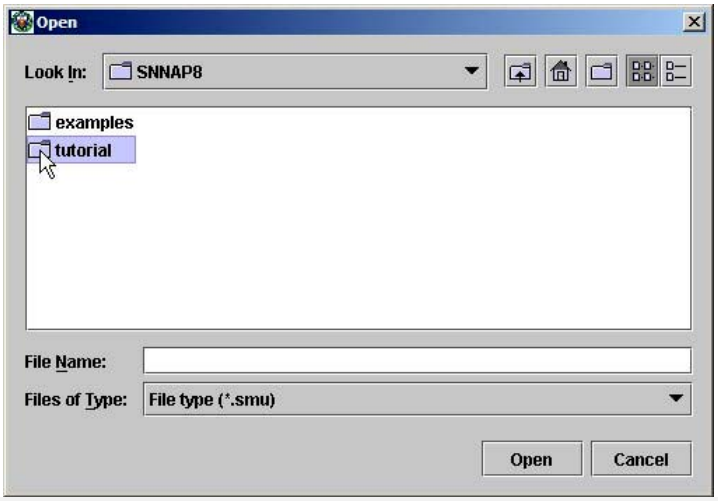

Figure 3.4: File chooser window.

The file chooser window opens at the top level of the SNNAP directory, providing you with the available folders/directories.

Double-click the folder icon next to the folder named `tutorial`. The file chooser will open the tutorial folder, double-click the folder named `tutorialExamples`. The folder tutorialExamples contains the example folders for this tutorial, one folder per chapter. Double-click the folder named `hhModel`.

Open the file `hh.smu` (either click once and click the `Open` button, or double-click the file icon).

The file will be loaded into the simulator.

Note: If your model is a large file, the loading process may take several seconds. Once the left hand buttons are enabled, they will display their title in black. If the button titles are in grey, they are not functional.

Click the `Start` button to begin running the simulation.

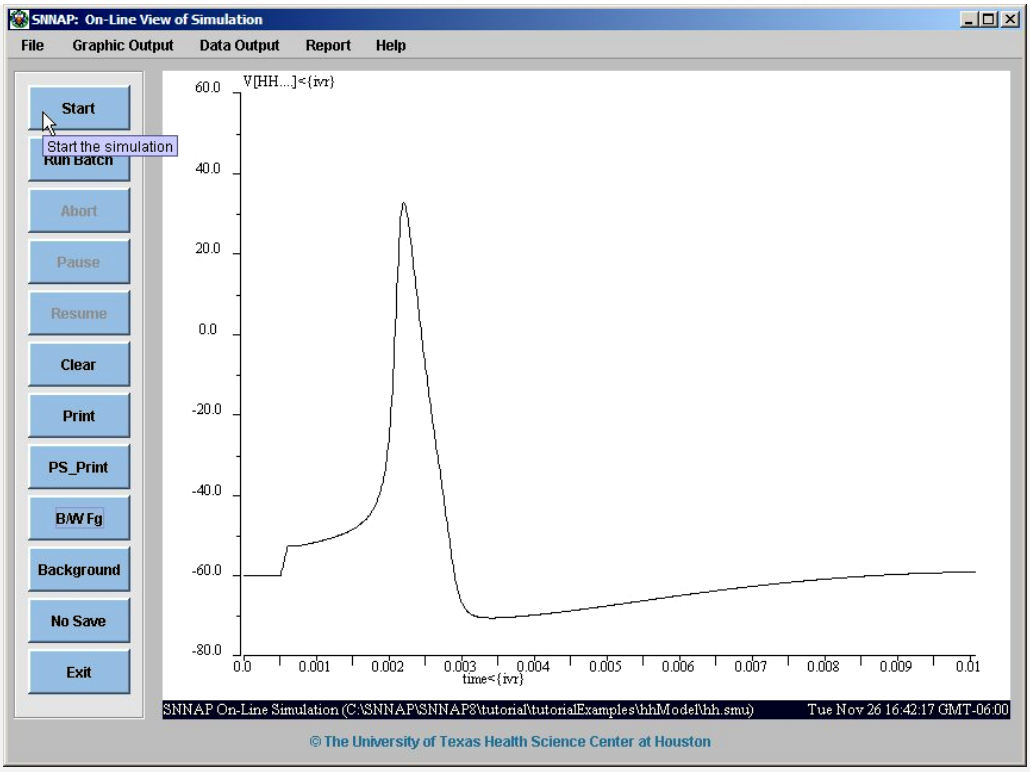

Figure 3.5. Single action potential in response to transient stimulation. Parameter as in Hodgkin and Huxley (1952) (with $V=V_{HH}$-60): C_m=1uF/cm^2, $\bar{g}_{Na}$ =120mS/cm^2, $\bar{g}_K$ =36mS/cm^2, $\bar{g}_L$ =0.3mS/cm^2, V_{Na} =55mV, V_K

Figure 3.5 shows an action potential which was initiated at time 0.0005 seconds (0.5ms) by a current injection of 75uA/cm^2 for a duration of 0.1ms. The action potential spike lasted for approximately 1 ms and at 0.01 seconds (10ms) the membrane potential was approaching the stable resting potential of -60mV.

Printing Simulation Results

To print the simulation results it is best to first change the color image to black on a white background. Select the button on the left pane of the viewing window. This button toggles the display between color and black/white. Now, select the `Print` button and your normal print window should open.

Once the image has been changed to black on white, you can restore the black background by selecting the button `Background`. A color coded window will open, and you may select any background color you want (the original was black).

Note: For the rest of this tutorial we will use a white background for the simulation window.

CHANGING SIMULATION PARAMETERS

So far, we saw how to launch and view a simulation. We will now learn how to change simulation parameters. First we will add an additional current injection to cause a second action potential during the simulation. Then we will edit model parameters, the maximal potassium conductance, to create a neuron model that exhibits oscillatory behavior with no external stimulation.

Adding Current Injection

Lets change parameters and rerun the simulation.

The first parameter change will be in the treatment file, which contains the various modes of applying stimulation to the model.

We will add a second stimulating current at time 7.5 ms.

In the main SNNAP window, click the button `Edit Treatment`. In the Treatment window click the menu item *File>Open*. In the file chooser window, double-click the file `hh.trt` (or click once and click the `Open` button).

The open treatment window should look as follows:

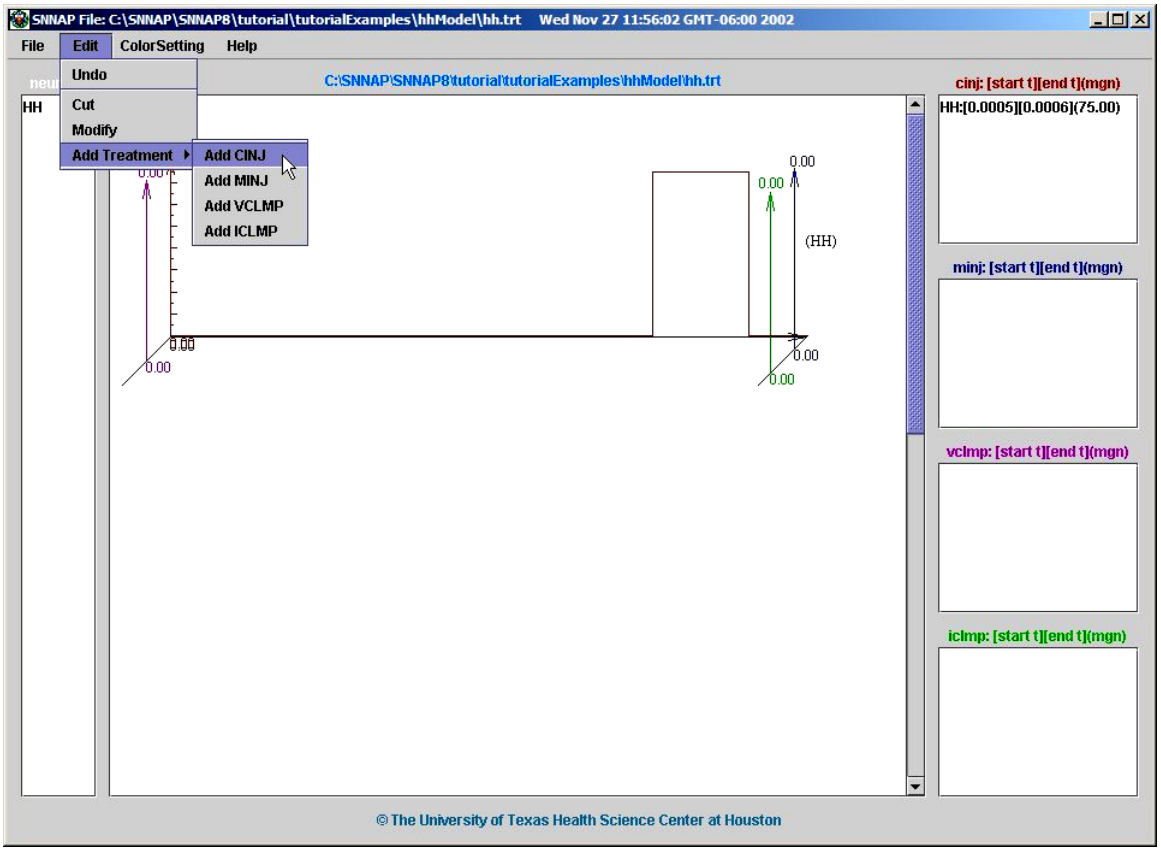

Figure 3.6. Treatment window with single current injection

The window shows the time axis and the various types of stimulation that may be applied to the model. We will add a second current injection.

Click *Edit>Add Treatment>Add CINJ* as seen in Figure 3.6.

The `Add CINJ` dialogue box will open. Enter the values as shown in the Figure 3.7:

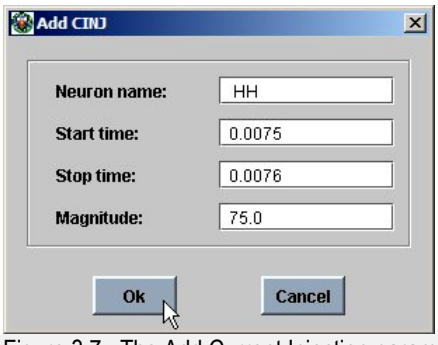

Figure 3.7. The Add Current Injection parameter window.

And click the button `Ok` to close the window.

The current injection stimulation we have entered is 75*uA/cm*^{2} for a duration of 0.1ms, starting at 7.5ms (0.0075 sec).

To rerun the simulation, click the menu *File>Reload Simulation* and then click the button `Start`. (If you have previously closed the simulation window, you must open it and load the `hh.smu` file, as was described in Running a Simulation).

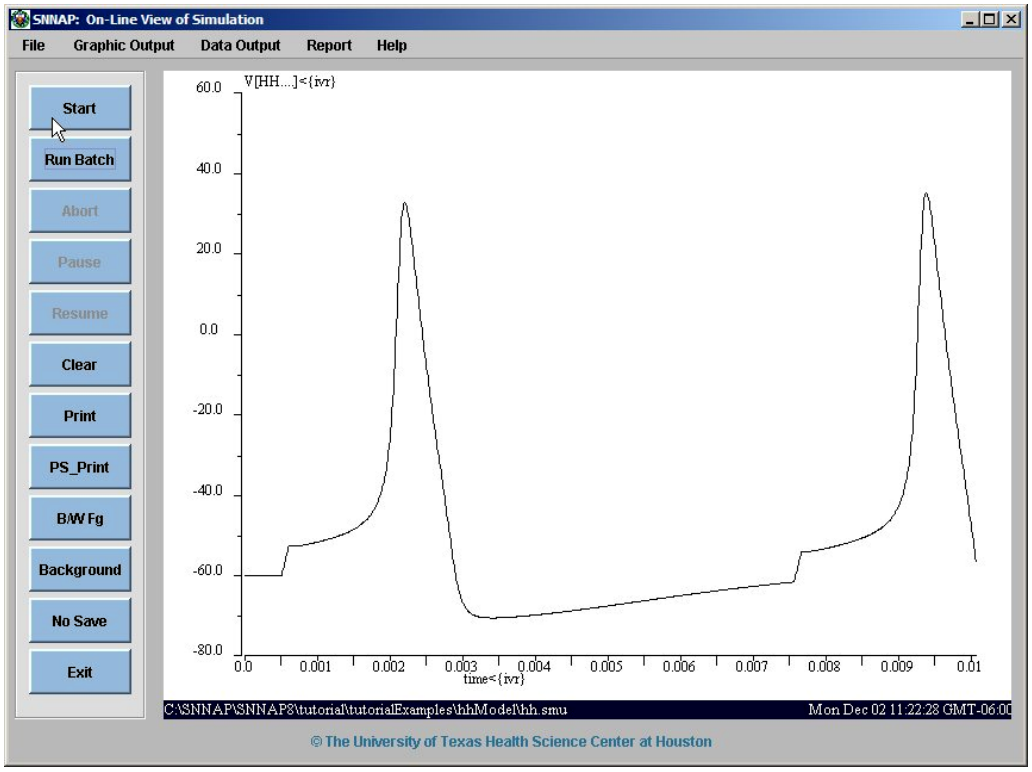

Figure 3.8. The simulation window displays two action potentials initiated by current injection at 0.5 and 7.5 ms.

Exercise – Changing the time of the second current injection, what is the shortest delay between two action potentials you can find.

CHANGING MODEL PARAMETERS

Up till now, we have seen how to add current injection to the model. Now we will look at how to change model parameters which alter the behavior of the model. The standard Hodgkin-Huxley model has a resting potential of approximately -60mV. This is due to the dynamics between the two major ionic currents, sodium and potassium. If we

reduce the maximal ionic conductance of the potassium current, $\bar{g}_K$, the model will show a higher resting potential. For some value of $\bar{g}_K$, the model will begin to exhibit oscillatory behavior. This value turns out to be 16*mS/cm*^{2}.

How do we do this?

First lets turn off the current injection by setting the magnitude to zero, and see that the HH model has a stable resting potential.

Open the Treatment window (as described in Section Add Current Injection), click the first injection data on the right and click the menu *Edit>Modify*. The `Modify Connection` dialog box will open. Set the current magnitude to zero. Repeat this for the second injection current as well. Then click the menu *File>Save* to save the changes you have made.

Go to the Simulation View window and click *File>Reload Simulation*.

And then click the button `Start`.

The display should be a stable membrane potential at approximately -60mV.

Changing Model Conductance Parameter

To change model parameters, click the button `Edit Formula` in the main SNNAP window, to open the formula editor.

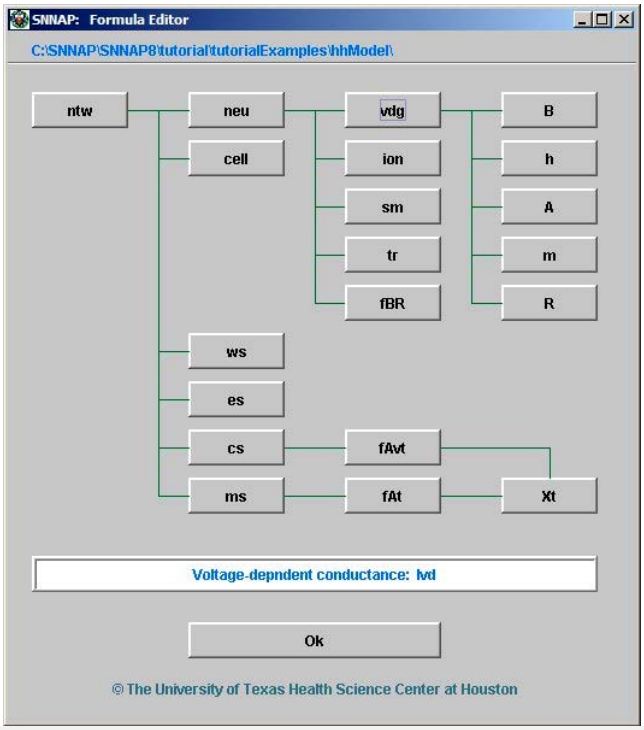

Figure 3.8. Formula Editor provides you with buttons to change model parameters.

The Hodgkin-Huxley model is a neuron model that uses voltage gated channels. In order to edit such channels, click the button `vdg`. The file chooser window will open. Click on the folder named `tutorial`, followed by the folder named `tutorialExamples`, and finally on the folder named `hhModel`. The model contains three channels, we are interested in the potassium channel, so either double-click on the `hhK.vdg` file, or click once and then click the button `open`.

This will open the window `Editor for *.vdg File`.

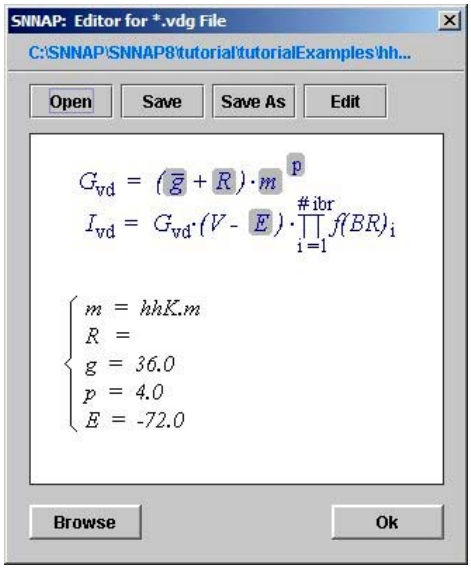

Figure 3.9. Editor for voltage dependent currents.

The parameters that you may change are marked by a grey background. Click on the maximal conductance parameter $\bar{g}$.

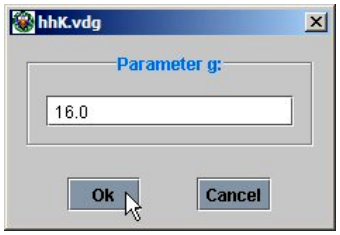

Figure 3.10. Window for modifying maximal conductance parameter.

Change the value of 36 to 16 and click the button `Ok`. Click the button `Ok` on the window `Editor for *.vdg File` and click `Yes` on the `Saved Modified File` question box that will appear.

Now you are ready to run the simulation with the new potassium conductance value.

Go to the window `Simulation View`, click *File>Reload Simulation* and then click the button `Start`.

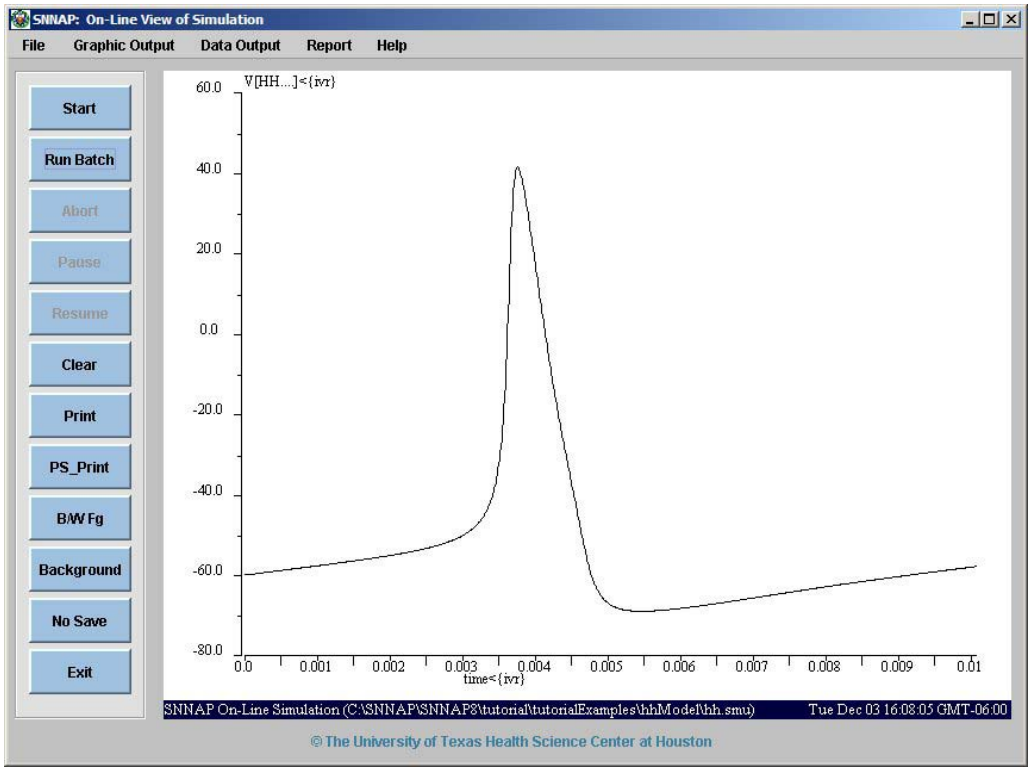

Figure 3.11 Oscillation with low maximal potassium conductance displaying a single action potential.

Figure 3.11 shows a single action potential. We would like to extend the time duration of the simulation in order to observe several action potentials. Lets see how to do that next.

Changing Simulation Duration

It would be helpful to see the oscillatory behavior of the model for a longer period than 10ms. To change the duration of the simulation there are two SNNAP windows that need to be altered.

In the main SNNAP window, click the button `Edit Simulation`.

In the `Edit Simulation` window click menu *File>Open*, and then click `HH.smu`, and the button `Open`.

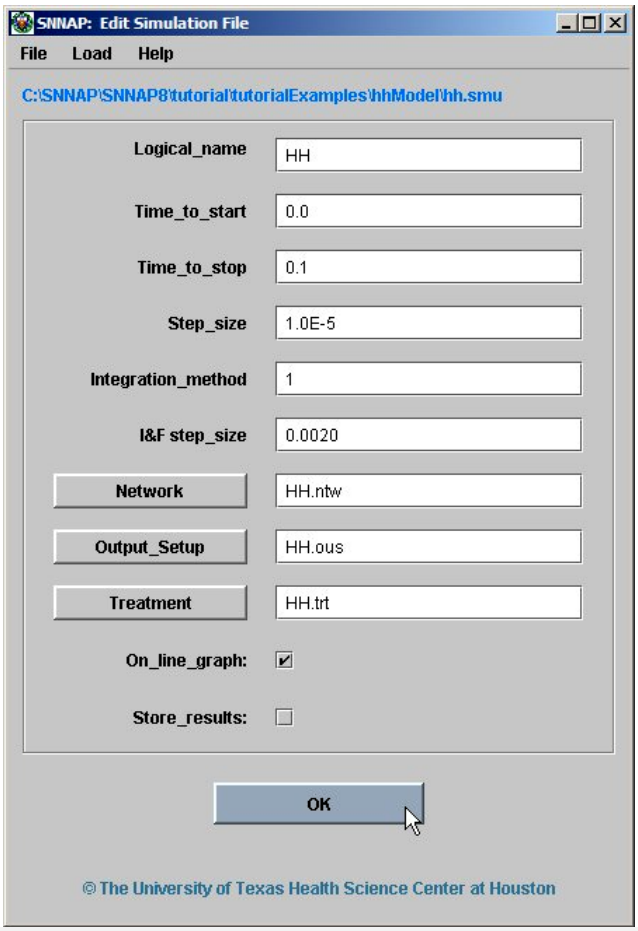

Figure 3.12 Window to edit simulation parameters.

Change the entry `Time_to_stop` to 0.1 seconds (100ms).

Click button `Ok`.

Click button `Yes` in the window `Saved Modified File`.

The simulation will now run for 0.1 seconds, rather than 0.01 seconds as before.

But if you rerun the simulation now, you will not see any different in the output window. The output window also needs to be changed to reflect the new simulation time duration.

Changing Output Display

The output display is controlled through the `Edit Output Setup` window.

Click the button `Edit OutScreen` in the main SNNAP control window to open the `Edit Output Setup` window, see Figure 3.13.

Open the `hh.ous` file that is in folder `hhModel`.

Click on the time variable in the channels pane on the left, which will highlight the selected variable with a blue background.

Click menu *Edit>Modify* to open the `Modify Channel Property` window.

In the `Modify Channel Property` window, change `maximum` value to 0.1, and number of ticks to 10.

Click the `Ok` button.

Select menu *File>Save* to save the parameter changes you have made.

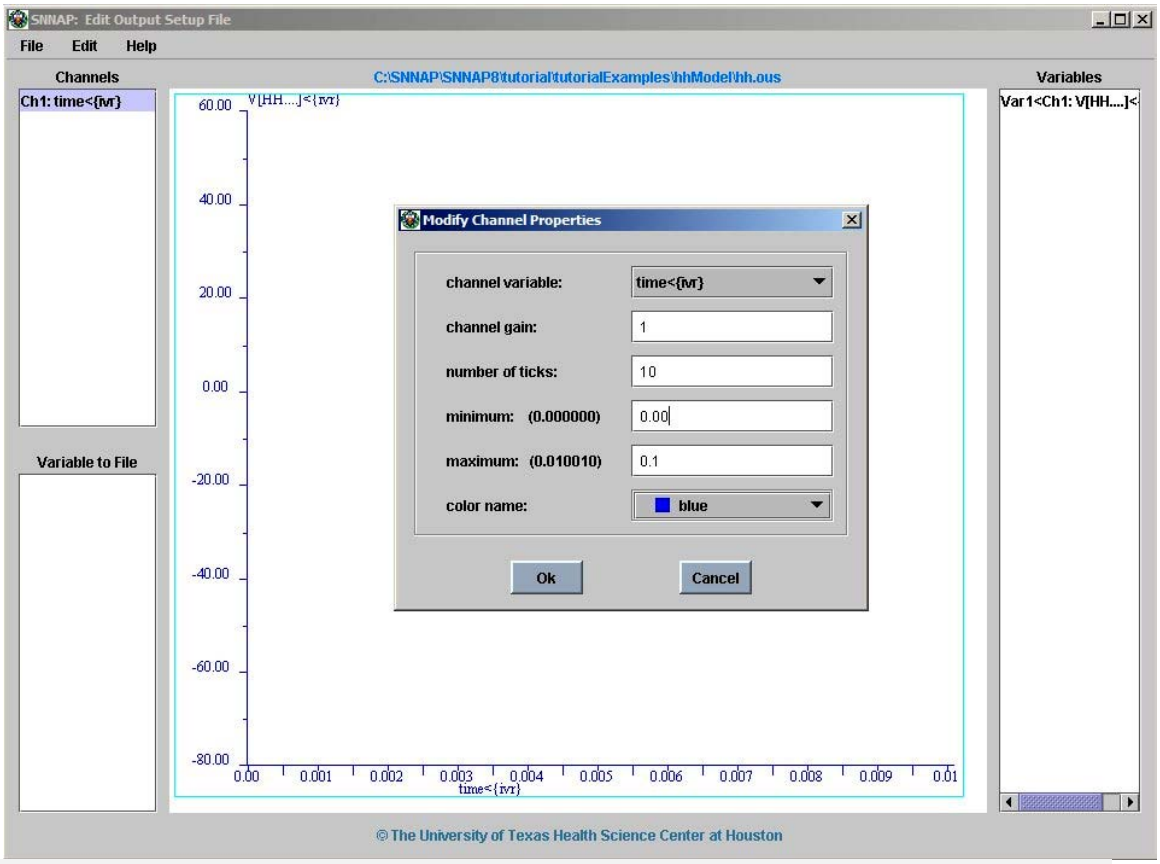

Figure 3.13 Window to edit output screen parameters with time axis modification dialog window overlay.

Now you can rerun the simulation and see the oscillator behavior of the Hodgkin-Huxley model with reduced potassium conductance.

Click *File>Reload Simulation* in the window On-Line View of Simulation, and then click the button Start.

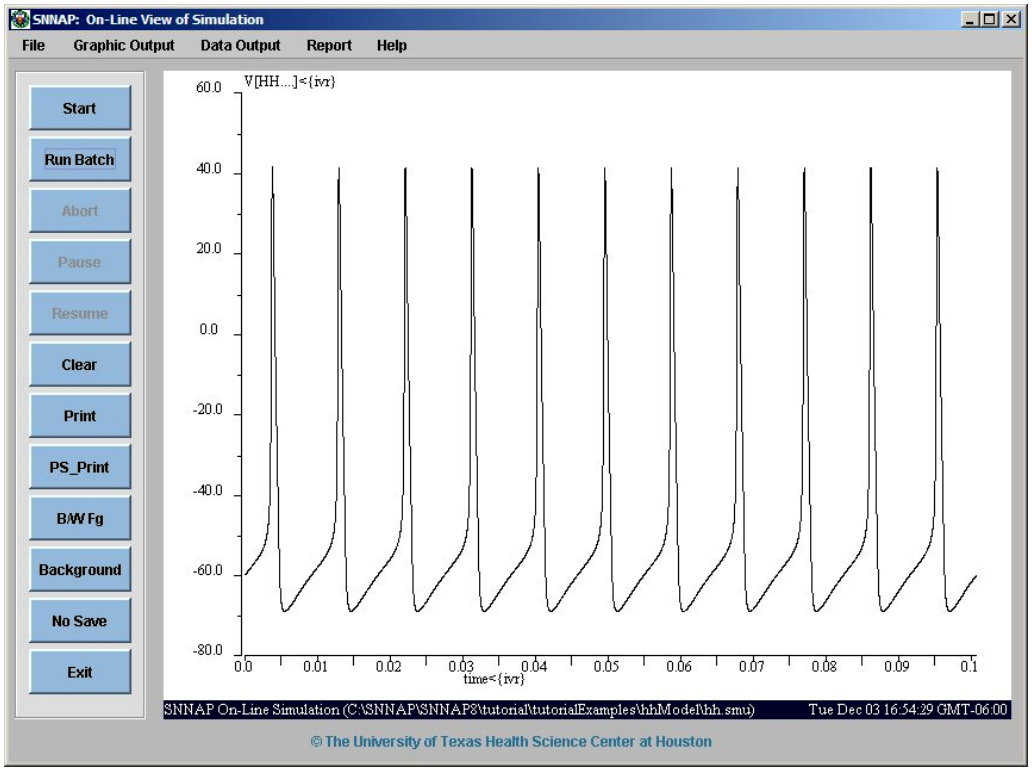

Figure 3.14 Hodgkin-Huxley model oscillations with reduced maximal potassium conductance.

Exercise – Starting from the original Hodgkin-Huxley values, the model can exhibit oscillatory behavior if the sodium conductance is increased. Can you find the minimum maximal sodium conductance to achieve oscillatory behavior?

CHAPTER 4: BURSTING NEURONS AND CENTRAL PATTERN GENERATORS

Many neurons show cyclic behavior of repetitive spiking activity followed by a period of inactivity. Such behavior is called bursting. Numerous types of cells exhibit bursting behavior, for example central pattern generator (CPG) neurons in invertebrates, thalamic neurons in mammals and pancreatic beta cells.

In this chapter we will develop a neuron model which exhibits bursting behavior, starting with a single spiking neuron model. You will learn how to incorporate new ion currents to a model as well as intracellular second messengers.

This chapter is divided into the following sections:

Introduction to Bursting Neurons – Neurons which exhibit periods of high-frequency activity followed by inactivity play an important role as central pattern generators.

The Morris-Lecar Spiking Neuron Model – The Morris-Lecar neuron model is a minimal biophysical model which exhibits single action potential.

Adding Ionic Currents - How to add additional currents to a neuron model in SNNAP.

Adding Ion Pools– How to add intracellular ions to a neuron model in SNNAP.

Adding Output Graphs –How to overlay output displays with SNNAP.

Central Pattern Generator Modulation – Multiple model parameters allow modulation of bursting behavior.

INTRODUCTION TO BURSTING NEURONS

Many invertebrate as well as mammalian neurons are bursting cells. They exhibit alternating periods of high-frequency spiking behavior (e.g. 100Hz) followed by a period with no spiking activity (quiescence period). The rich dynamic behavior of such neurons attracted both neuroscientists and mathematicians in an effort to understand the underlying mechanisms of bursting behavior and its modulation.

There are numerous neural circuits that contain pacemaker neurons, which provide a driving force for the network. Such pacemaker neurons contain bursting mechanisms which endow them with the ability to fire independently of external stimulation. Other neurons have bursting ability which may require a transient input to initiate their burst cycle. Central pattern generators may be of either pacemaker type (endogenous burster) or requiring a transient input to initiate a burst (conditional burster).

Neuronal bursting behavior may result from various mechanisms. One ubiquitous mechanism leading to bursting is the oscillation of a slow calcium wave that depolarized the membrane and causes a series of action potentials on top of the calcium wave. When the slow wave ends, the quiescent period begins. Such calcium waves have also been implicated in the rhythmic behavior of cardiac muscles.

In this chapter we will develop a bursting neuron model. Starting with a neuron model that exhibits single action potentials, we will convert this model to a neuron model that exhibits bursting behavior by incorporating calcium-dependent processes which will allow for the formation of a slow calcium wave.

THE MORRIS-LECAR SPIKING NEURON MODEL

The Morris-Lecar model was developed to describe the behavior of barnacle muscle cells. The model exhibits single spiking behavior, similar to the Hodgkin-Huxley model. The Morris-Lecar (ML) model though, is simpler than the Hodgkin-Huxley (HH) model in that it consists of only two variables, the membrane potential (*V*) and potassium

activation (*w*), rather than four variables. The two ordinary differential equations are described as follows:

$$C\frac{dV}{dt} = I - \bar{g}_{Ca} m_{\infty}(V)(V - V_{Ca}) - \bar{g}_{K} w(V - V_{K}) - \bar{g}_{L}(V - V_{L})$$

$$\frac{dw}{dt} = \phi \frac{[w_{\infty}(V) - w]}{\tau_{w}(V)}$$

where the three ionic currents, calcium (Ca), potassium (K) and leak (L) are products of a maximal conductance $\bar{g}$, activation component (function m_{∞}, variable *w*, or constant value 1, respectively) and a driving force (for further detail see Section Hodgkin-Huxley Model).

The potassium activation variable *w* is defined by a steady-state activation function w_{∞} (a sigmoid curve) and a time-constant function τ_{w} (a bell shaped curve).

$$w_{\infty}(V) = \frac{1}{1 + e^{(V - h_w)/-s_w}}$$

$$\tau_{\infty}(V) = \frac{1}{e^{(V - h_w)/(2*s_w)} + e^{-(V - h_w)/(2*s_w)}}$$

The calcium current also uses a steady-state activation function m_{∞}, which is defined as:

$$m_{\infty}(V) = \frac{1}{1 + e^{(V - h_m)/-s_m}}$$

Simulation of the Morris-Lecar Model

The simulation of the Morris-Lecar model is similar to running the Hodgkin-Huxley model, see Section Running a Simulation in Chapter 3.

Open the main SNNAP window

Click the button `Run Simulation` to open the simulation window `On-Line View of Simulation`, as shown in Fig. 4.1.

Click the menu *File>Load Simulation*.

Locate and select the file SNNAPhome\SNNAP8\tutorial\tutorialExamples\bursterCPG\ml.smu in the file chooser dialog box and click `Open`.

Click the button `Start`

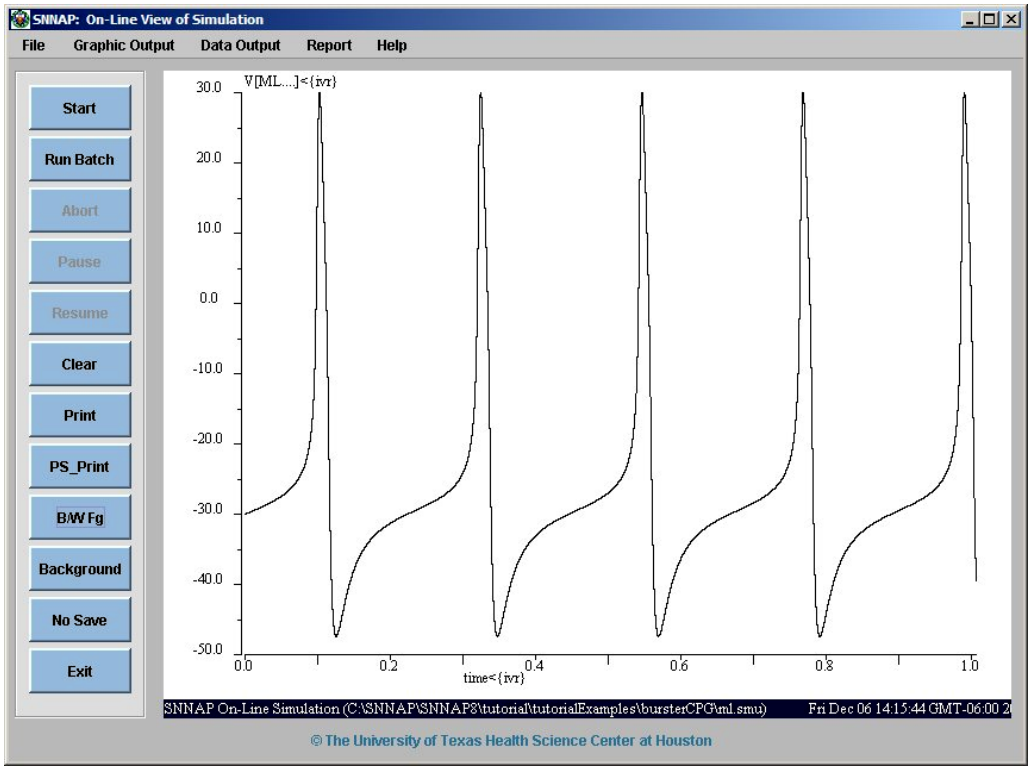

Figure 4.1. Simulation of the Morris-Lecar model. Parameters are: C_m=20*uF*/cm^2, $\bar{g}_{Ca}$ =4.0mS/cm^2, $\bar{g}_K$ =8.0mS/cm^2, $\bar{g}_L$ =2.0mS/cm^2, V_{Ca} =120mV, V_K =-84mV, V_L =-60mV, ϕ = 1/15, h_m =-1.2, s_m =9, h_w

Description of the Morris-Lecar Model

The Morris-Lecar model consists of three currents, a calcium current, a potassium current and a leak current. The calcium current is the inward current, similar to the sodium current in the Hodgkin-Huxley model. The activation of the calcium current is a fast process, in comparison to the activation of the potassium current and is therefore

modeled as instantaneous, using the steady-state activation function rather than a time-dependent variable. This means that for any value of voltage *V*, the steady state function $m_\infty(V)$ is calculated. This calcium current does not incorporate any inactivation process. The potassium current has an activation variable *w*, similar to the Hodgkin-Huxley variable *n*, though with no power exponent. Finally, the leak current is similar to the HH model.

Note: The original Morris-Lecar model described the activation and time constant functions using hyperbolic tangent and cosine functions. We have converted those functions to the more commonly used sigmoidal functions using Boltzmann-like exponential functions.

ADDING IONIC CURRENTS

We will now add a new ionic current to the Morris-Lecar model, a calcium-dependent potassium current. Such currents are known to play a role in bursting neurons, providing a mechanism for ending the bursting period and influencing the duration of quiescence. In the main SNNAP window, click the button `Edit Neuron`. This will cause SNNAP to display the `Edit Neuron` window. Next click *File>Open* to open the file chooser dialog box. The file we need is `ml.neu` in the folder *SNNAP8\tutorial\tutorialExamples\bursterCPG*. Click the file name to select and click the button `Open`. Figure 4.2 shows the resulting display of the `Edit Neuron` window. Next, click the menu *Edit>Add Node>V.D. Conductance*, to open the `Add Node` dialog box.

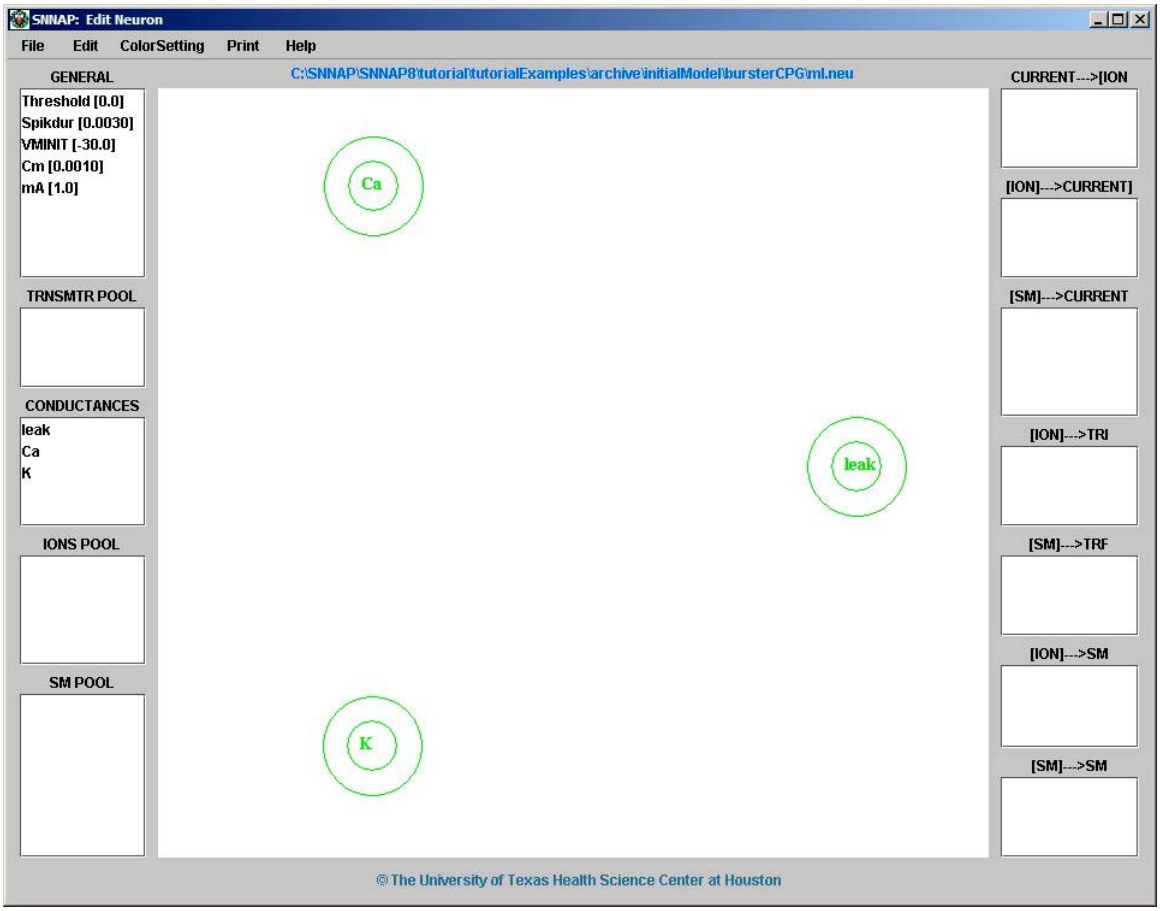

Figure 4.2. Use the window Edit Neuron to incorporate new conductances.

In the Add Node dialog box, you must provide the name of the conductance, *K(Ca)*, and the file name, *mlKCa.vdg*, which will contain the parameter values, as shown in Fig. 4.3. Click Ok to close the dialog box and update the model.

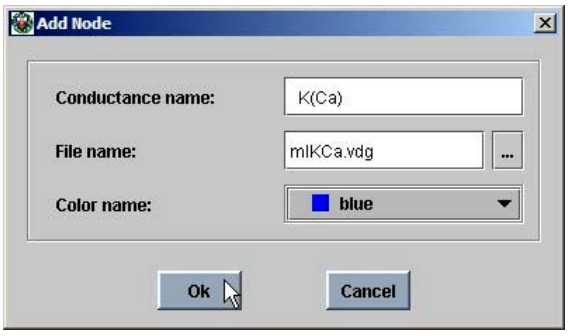

Figure 4.3. Add Node dialog box to add new conductance to model.

The next step is to establish the equations for this new conductance.

But first we must create a file named `mlKCa.vdg`. SNNAP is provided with a formula template library, folder `formulaTemplates`, where you can access all the equations that may be used with SNNAP.

In the main SNNAP window click button `Edit Formula`.

In the window `Formula Editor` click the button `vdg`. The file chooser dialog box will be displayed. The template file we need is `default.vdg` in the folder *SNNAP8\tutorial\formulaTemplate*. Click on the file name to select it, and click the button `Open`. The window `Editor for *.vdg File` will be displayed as in Fig. 4.4. Click the *Edit* button to display the menu of equations.

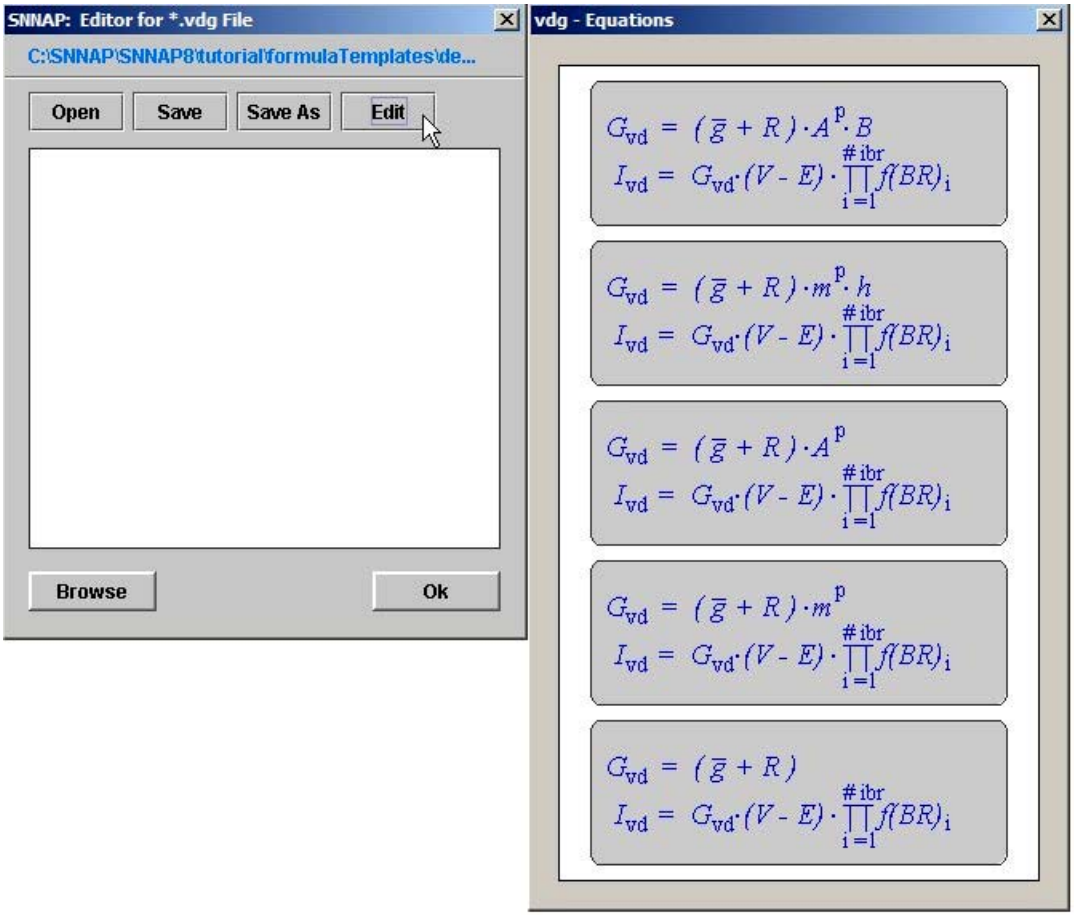
SNNAP: Editor for *.vdg File
C:\SNNAP\SNNAP8\tutorial\formulaTemplates\de...
Open
Save
Save As
Edit
Browse
Ok
vdg - Equations
$G_{vd} = (\bar{g} + R) \cdot A^{p} \cdot B$
$I_{vd} = G_{vd} \cdot (V - E) \cdot \prod_{i=1}^{\#ibr} f(BR)_i$
$G_{vd} = (\bar{g} + R) \cdot m^{p} \cdot h$
$I_{vd} = G_{vd} \cdot (V - E) \cdot \prod_{i=1}^{\#ibr} f(BR)_i$
$G_{vd} = (\bar{g} + R) \cdot A^{p}$
$I_{vd} = G_{vd} \cdot (V - E) \cdot \prod_{i=1}^{\#ibr} f(BR)_i$
$G_{vd} = (\bar{g} + R) \cdot m^{p}$
$I_{vd} = G_{vd} \cdot (V - E) \cdot \prod_{i=1}^{\#ibr} f(BR)_i$
$G_{vd} = (\bar{g} + R)$
$I_{vd} = G_{vd} \cdot (V - E) \cdot \prod_{i=1}^{\#ibr} f(BR)_i$

Figure 4.4. Voltage-dependent gated channel equations.

The menu of equations provides you with various conductance models to choose from. The first four equations models use voltage-dependent activation. Our calcium-dependent potassium current uses intracellular calcium concentration for activation. The last equation is needed for such a conductance.

Figure 4.5 shows the equation with the required parameter values.

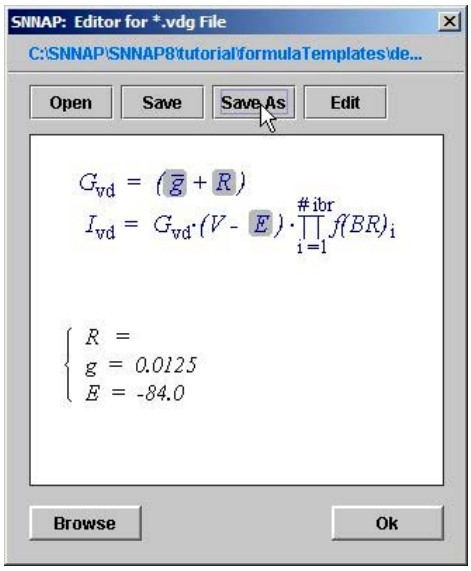

Figure 4.5. Equation for calcium-dependent potassium conductance in vdg editor.

There are two parameters which must be edited, the maximal conductance value $\bar{g}$ and the reversal potential *E*. Parameters which have grey backgrounds may be edited. To edit the value of such a parameter, point the mouse over the grey area and click. This will cause a dialog box to appear (similar to Fig. 3.10), permitting you to edit the value. After entering the desired value in the dialog box click the button `Ok`, to close the box. The value you have entered will now appear in the `*.vdg` window.

Once you are done editing the two parameters, click the button `Save As` in the `vdg` editor to open the file chooser dialog box, see Fig. 4.6. We do not want to save the equation file in the folder `formulaTemplates`, but rather in the appropriate folder so that SNNAP can locate the file when loading the simulation. The file name `mlKCa.vdg` was declared in the window `Edit Neuron` (see Fig. 4.2) and should be provided. The

path for the equation file is *SNNAPhome\SNNAP8\tutorial\tutorialExample\bursterCPG\mlKCA.vdg*.

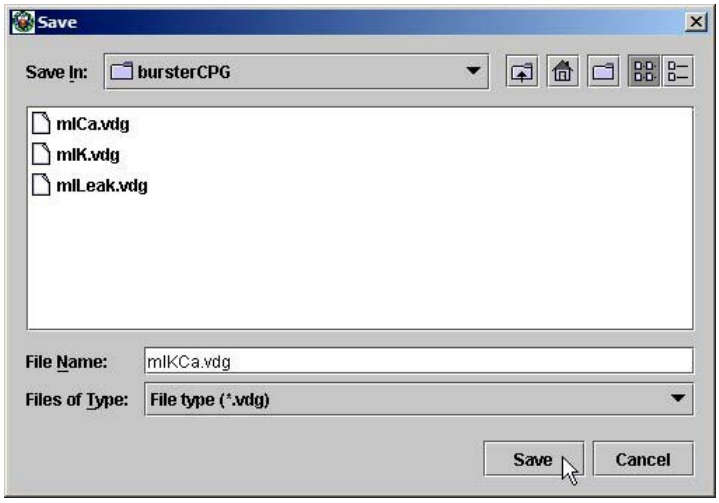

Figure 4.6. Dialog box to save vdg file in model folder.

We have now completed the description of the new ion conductance and incorporated it into SNNAP.

The use of calcium to activate a potassium current requires us to introduce the concept of calcium as an intracellular ion pool. We now turn to introducing how to use ion pools with SNNAP.

Ion Pools

Ions can play a role in a cell as molecules that activate or inactivate certain processes. Such ions are called *ion pools* in SNNAP. Calcium-dependent potassium channels are activated by intracellular calcium, the higher the calcium concentration the higher the channel activation. In order to use an ion as a second messenger, SNNAP must be programmed to calculate the ion concentration. Use the window `Edit Neuron` to declare ion pools in SNNAP.

To open the `Edit Neuron` window, click the button `Edit Neuron` in the main SNNAP window. Locate and select the file `ml.neu`, in the folder

SNNAP8\tutorial\tutorialExamples\bursterCPG and click the button Open. Figure 4.7 shows that the *K(Ca)* conductance was included into the model description.

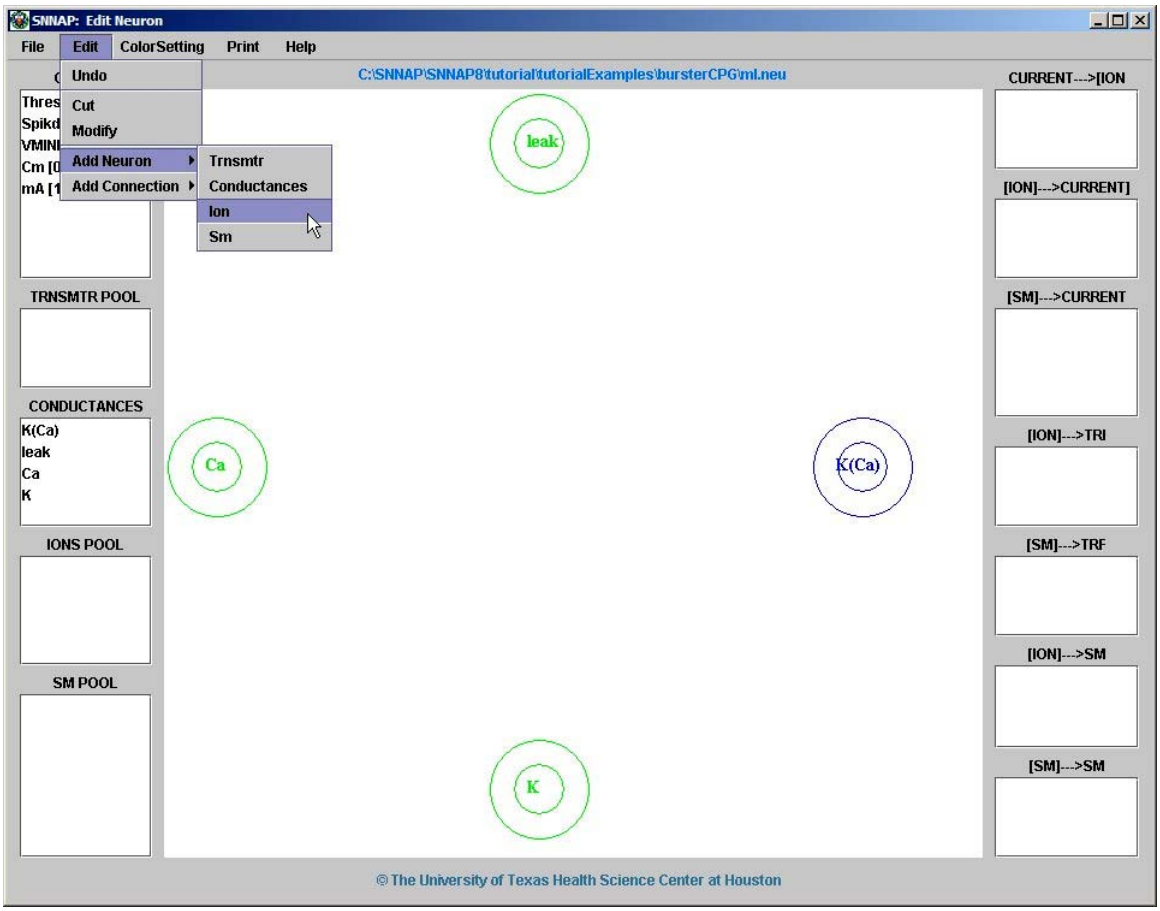

Figure 4.7. Adding an ion element in the Edit Neuron window.

To add an ion pool, click *Edit>Add Node>Ion*, which will open an Add Node dialog box. Figure 4.8 shows the dialog box with the corresponding parameter values.

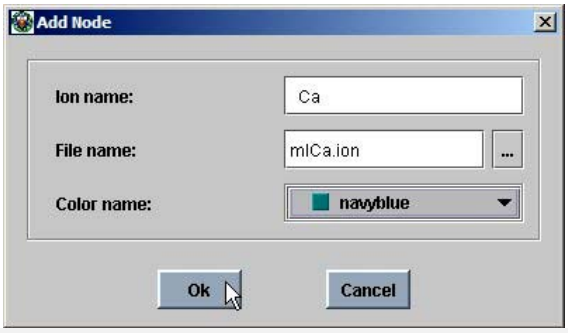

Figure 4.8. Declaring a new ion in the model.

Edit the dialog box and enter the name of the ion (*Ca*) and the file name which will contain the equation for this ion (`mlCa.ion`). Choose a color for the display of the ion, and click `Ok`.

The `Edit Neuron` window will be updated to display *Ca* in the `IONS POOL` pane (left side), and the an icon with *Ca* in the color you selected will be displayed in the center pane (see Fig. 4.9).

The next step is to connect the processes that will be using this ion pool. The calcium current will be the source of *Ca* ions, and the ion concentration will activate the current *K(Ca)*.

To indicate to SNNAP that a current is an ion source, click *Edit>Add Connection>Current_to_Ion* in the `Edit Neuron` window, as shown in Fig. 4.9.

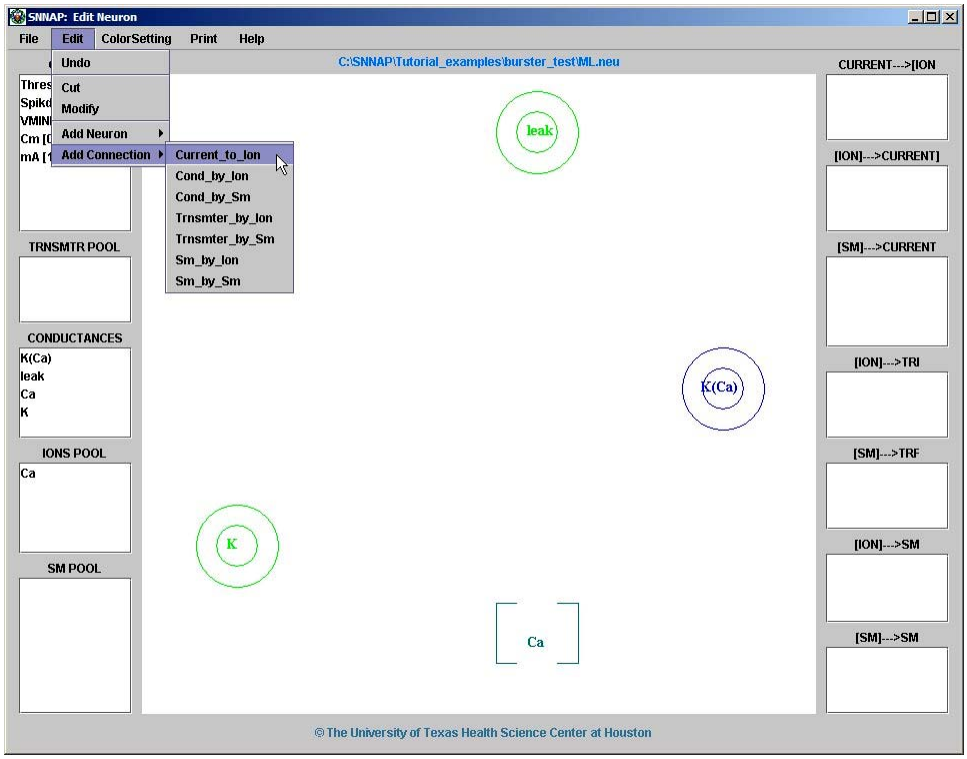

Figure 4.9. Converting current to ion concentration in Edit Neuron window.

The `Add Connect` dialog box will be displayed, see Fig. 4.10. Set the `Conductance Name` to *Ca*, by using the pull-down menu, click in the box to the right of the text `Conductance Name`. Since there is only one ion declared in SNNAP, the `Ion name` is correctly set.

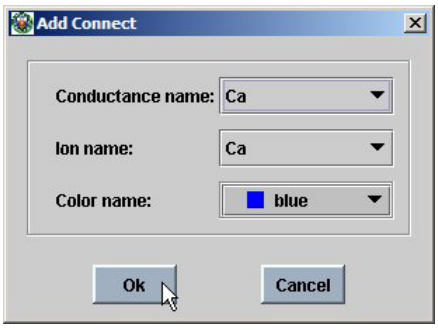

Figure 4.10. Select current source for ion.

Click the button `Ok` when you are done (you may also choose a color using the pull-down color menu).

An arrow will be displayed in the `Edit Neuron` window connecting the *Ca* conductance icon (circle) and the *Ca* ion icon (brackets), see Fig. 4.12.

To set the current which will be affected by the ion, click `Edit>Add Connection>Cond_by_Ion` in the `Edit Neuron` window. The `Add Connect` dialog box will appear, as seen in Fig. 4.11. The equation which governs the modulation of the current using the designated ion will reside in the file that is declared using this dialog box. Set the filename to `mlCa.fBR`.

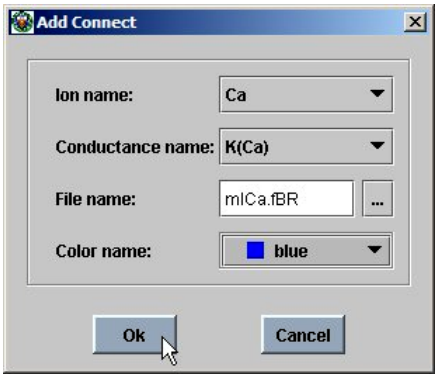

Figure 4.11. Select the destination current for ion activation, and provide the filename which will contain the governing equation.

Click the button `Ok` in the dialog box. The final display of the `Edit Neuron` window is shown in Fig. 4.12.

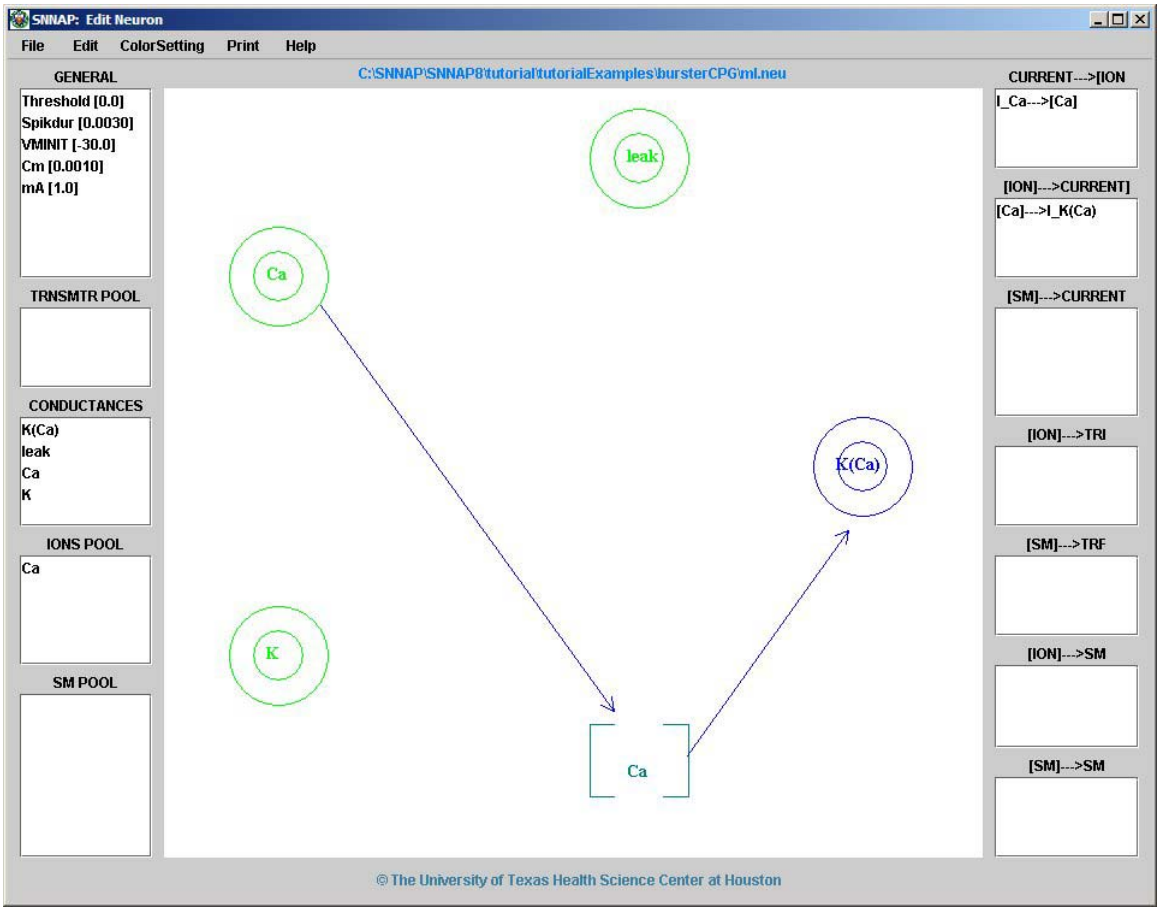

Figure 4.12. Final view of Edit Neuron window with incorporated calcium ion.

On the right hand side of the window, the pane `Current → Ion` indicated that the calcium current is the source of *Ca* ions, while the pane `[ion] → current` indicates that the *Ca* ions have an affect on the calcium-dependent potassium current.

Click `File>Save` in the `Edit Neuron` window to save the changes to the neuron model in SNNAP.

We have now programmed SNNAP to describe an ion process where calcium ions accumulate due to the *Ca* current and play a role in the mechanism of the calcium-dependent potassium conductance.

What remains now is to establish the exact equations for these two processes.

Setting Ion Equation

SNNAP uses numerous equations when running a simulation. The `Formula Editor` window allows you to access all the equations used in a model. In the main SNNAP window, click the `Edit Formula` button to open the `Formula Editor` window (see Fig. 3.8).

In the previous section, we named two files (`mlCa.ion` and `mlCa.fBR`) which we now will establish. Using the template library, we will choose the equation for each process.

Click the button `ion` in the `Formula Editor` window. In the file chooser window go to the template libraries in SNNAPhome`\SNNAP8\tutorial\formulaTemplates.`

The file `default.ion` will appear. Click the file name and click `Open`. The window `Editor for *.ion File` will appear with an empty window. Click the button `Edit`, and the list of possible ion equations will appear.

Click on the second equation, and it will appear in the Editor window as shown in Fig. 4.13.

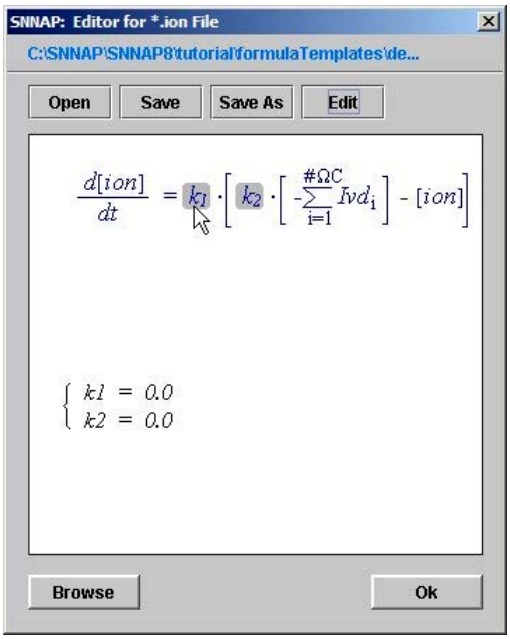

Figure 4.13. Equation for intracellular calcium ion concentration.

There are two parameters that need to be set, the parameter *k1* to 5.0 and parameter *k2* to 0.4. To set a parameter, click on the parameter marked with a grey background to open a dialog box where you set the appropriate value. Click the button Ok to close the dialog box. When all three parameters have the desired values, click the button Save As, in order to save the file under the name mlCa.ion in the desired folder. Remember that we declared that file name when updating the Edit Neuron window, see Fig. 4.8. In the Save dialog box, change the folder to \SNNAPhome\SNNAP8\tutorial\tutorialExamples\bursterCPG, and click the button Save. Once saved, close the Editor for ion File window by clicking the button Ok, or the upper right button x.

We will now turn to the second file that needs to be established, the fBR file. This file will contain the equation for the activation of the potassium conductance that is dependent on intracellular calcium concentration.

In the Formula Editor window click the button fBR to open the Editor of *.fBR File window. Click the button Edit to open the fBR-Equation window. Click the first equation, as seen in Fig. 4.14.

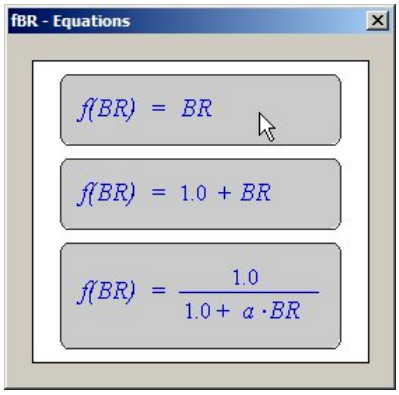

Figure 4.14. fBR equation window.

Clicking the first equation will place the equation in the Editor window, see left side of Fig. 4.17. Before continuing, it would be advantageous to save the file in the appropriate folder. Click the button Save As, which will open a file chooser dialog box, see Fig. 4.15. Traverse your folder structure to the folder bursterCPG, the complete path is: \SNNAPhome\SNNAP8\tutorial\tutorialExamples\bursterCPG.

Save the file in the correct directory under the name `mlCa.fBR` which was provided to SNNAP, see Fig. 4.11. Click the button `Save` to save the file and close the dialog box.

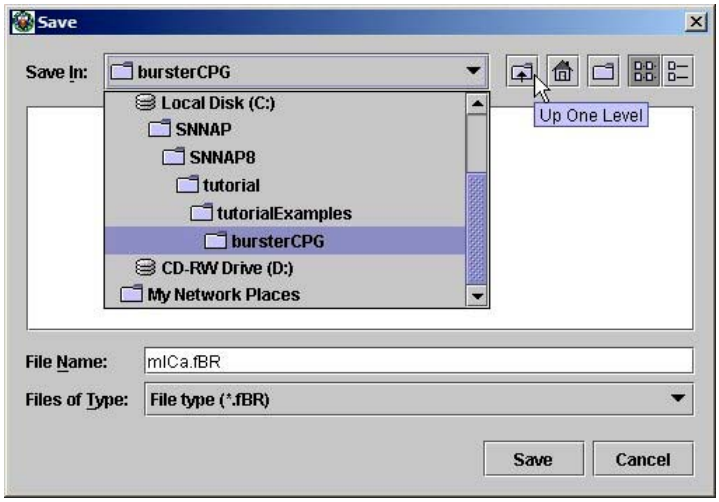

Figure 4.15. File chooser dialog box.

In the editor window, click the parameter `BR` with the grey background to open the `BR-Equation` window. Click the button `Edit` to display the possible equations, shown in Fig. 4.16.

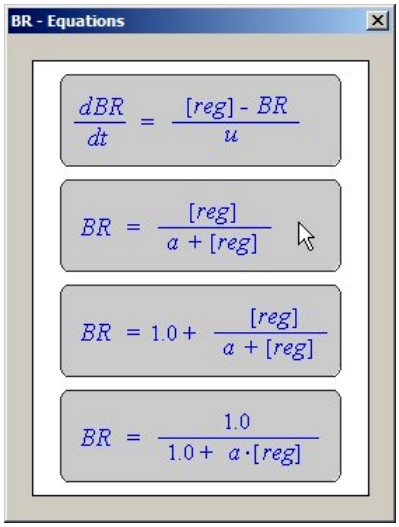

Figure 4.16. BR equation window.

Click the second equation (Mechaelis-Menten form) and it will appear in the `Editor` window, as shown in Fig. 4.17.

There is one parameter which must be set, click on parameter a (the disassociation constant) and set the value to 1.0.

Click the button `Ok` to close the window and click `Yes` in the `Save Modified File` dialog box.

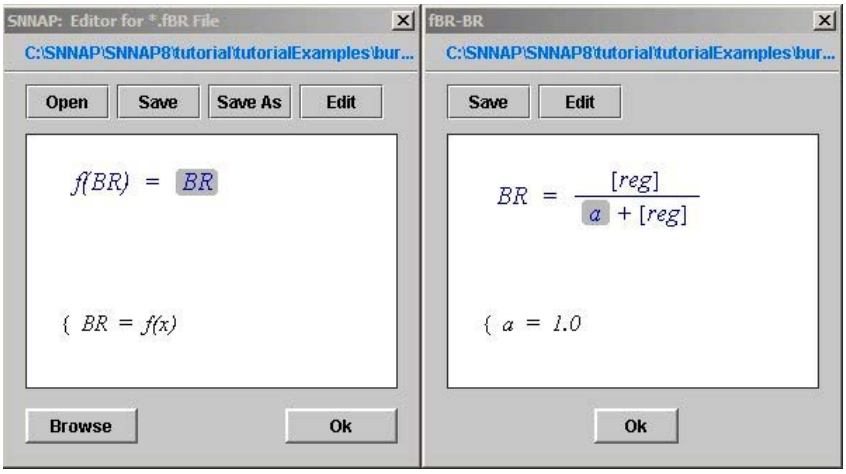

Figure 4.17. Biochemical regulation equation of Mechaelis-Menton form.

There are two parameters that must be changed for the model to exhibit bursting behavior, the maximal time constant, and the input current.

To change the time constant parameter you need to open the time constant equation in the appropriate editor. Open the `Formula Editor` window and click on button `A` to open the file chooser window. Locate the file `mlK.A`, in folder:

`\SNNAPhome\SNNAP8\tutorial\tutorialExamples\bursterCPG.`

Open the file `mlK.A` and click on parameter `tA`. A window will appear where you can click on t_{max}, see Fig. 4.18, to open a dialog box.

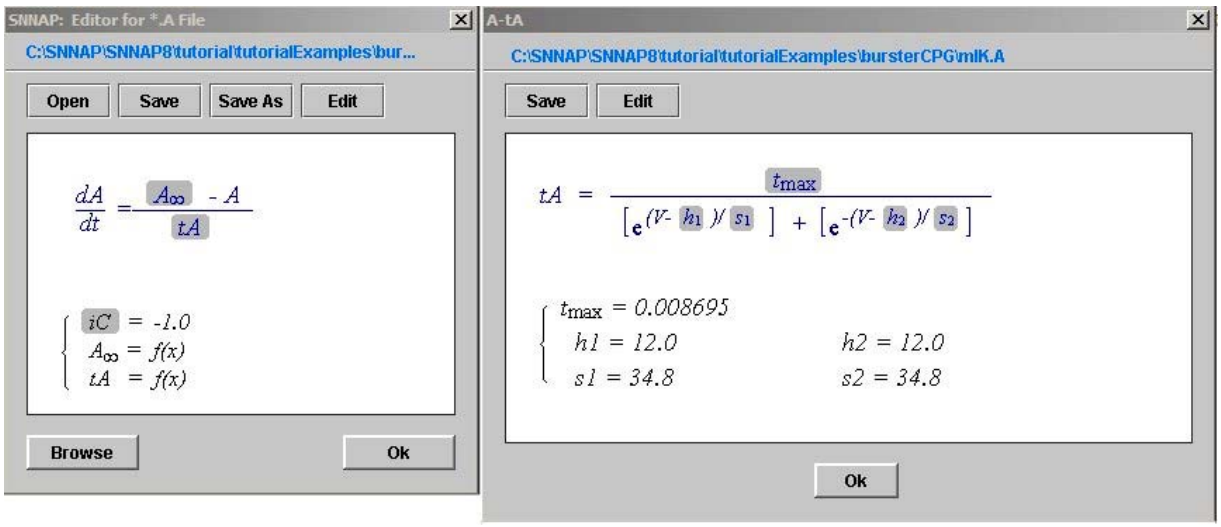

Figure 4.18. To change the maximal time constant the *A* file and *tA* file are edited.

Set the t_{max} parameter value to 0.008695 in the dialog box and click `Ok`. The parameter will be updated as shown in Fig. 4.18.

Click `Ok` to close the `A-tA` window, and click `Yes` in the `Save Modified File` window. Click `Ok` to close the `*.A file` editor window.

The input current parameter must be changed in the treatment file. In the main SNNAP window click on the `Edit Treatment` button. Follow the procedure in section Adding Current Injection (see also Fig. 3.6) and change the injection current to the value 2.25.

Now we can run our model and see the result.

In the main SNNAP window, chick on button `Run Simulation`. to open the simulation window. If the window is already open, click in the window to make it active.

Click on *File>Load Simulation,* locate and select the simulation file:
`\SNNAPhome\SNNAP8\tutorial\tutorialExampeles\busterCPG\ml.smu`

and click `Open`.

Click the button `Start` in the main SNNAP window.

The simulation display should resemble that in Fig. 4.19.

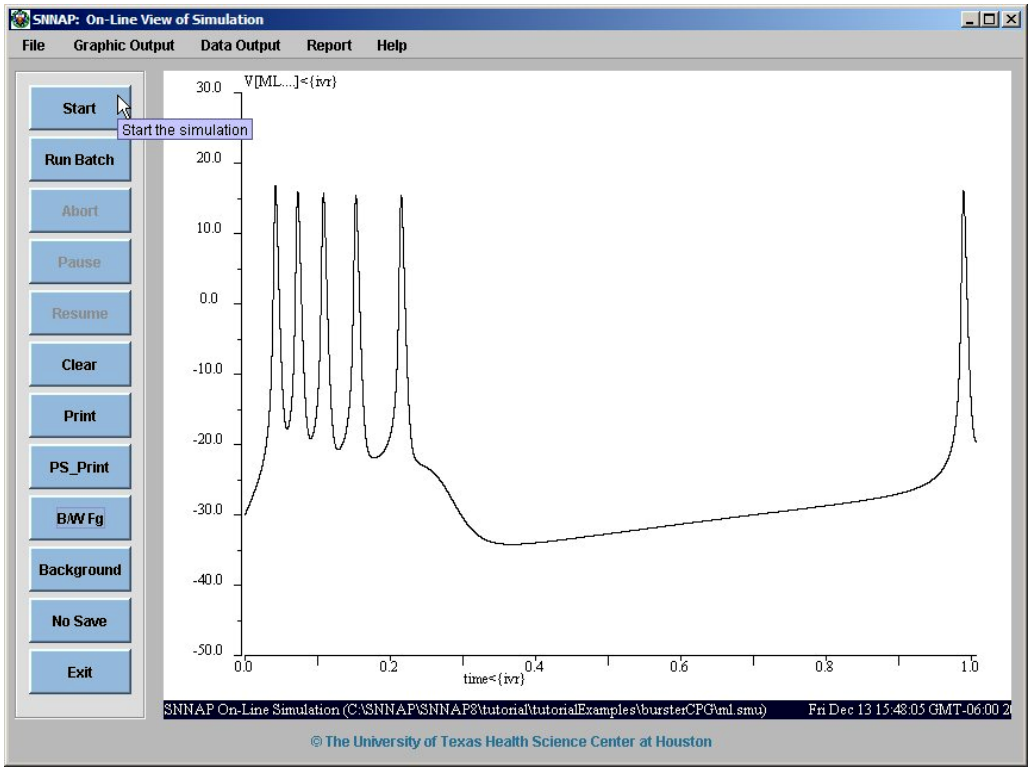

Figure 4.19. Bursting behavior of Morris-Lecar model with Rinzel-Ermentrout extensions.

Lets change the duration of simulation to three seconds in order to display several burst periods. This we will do in the next section, dealing with the output screen.

CHANGING OUTPUT SCREEN

There are numerous model variables and functions that can be displayed on the output screen.

But first lets extend the duration of simulation.

We have done this previously in Section Changing Simulation Duration.

Edit the simulation file `ml.smu` and extend the simulation to three seconds.

Edit the Output file and modify the time variable to a maximum time of three seconds.

Edit the treatment file and modify the duration of stimulation to 3 seconds.

You can rerun the simulation and observe several bursting periods.

Lets add the variable intracellular calcium concentration to the output screen and see the evolution of this variable with time.

Adding New Output Variable

To add a new output variable open the Output Screen window (click button `Edit OutScreen` in main SNNAP window). Click *File>Open* and select the file `ml.ous`.

In the `Edit Output Setup File` window click *Edit>Add>Add Variable*

A variable dialog box will appear. Edit the dialog box to resemble Fig. 4.20.

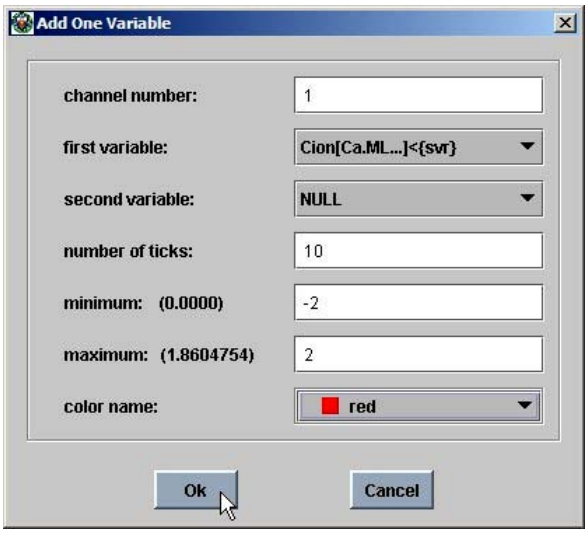

Figure 4.20. Declaring a second output variable.

The variable we are interested to display is the *Ca* variable of the ion equation. Note that the value can not decrease below zero, but in order to shift the display, we are using -2 as the lower bound.

After closing the variable window, by clicking `Ok`, save the output window (`File>Save`) and rerun the simulation.

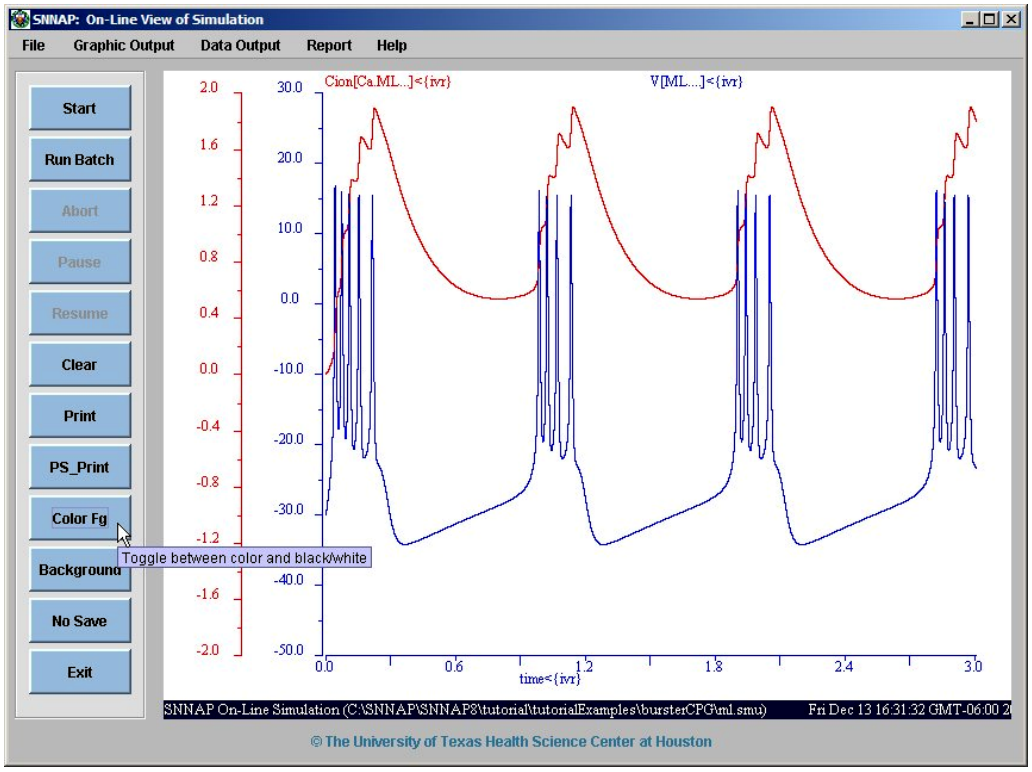

Figure 4.21. Bursting model displaying membrane potential (*V*) and intracellular calcium (*Ca*)

The model simulation of three seconds, displays three full burst periods. The calcium concentration oscillates as well, and is responsible for ending the burst at its peak value, by activating the calcium-dependent potassium current.

MULTIPLE OUTPUT GRAPHS

There are numerous system variables and functions we can display to aid in understanding how model behavior comes about. For example, it is clear that in our model, membrane voltage fluctuates between -50 and +15mV, while intracellular calcium concentration varies between 0.5 and 1.9uM. There are ionic currents which may vary by several orders of magnitude. It would therefore be convenient to view several graphs on one screen, using different scales.

As a first step, lets see how to separate the two displays in Fig. 4.21. Open the screen editor by clicking the `Edit OutScreen` button in the main SNNAP window. In the `Edit Output Setup File` window, click *Edit>Add>Add Channel* which will open the

Modify Channel Properties dialog box, as shown in Fig. 4.22, set the parameters accordingly.

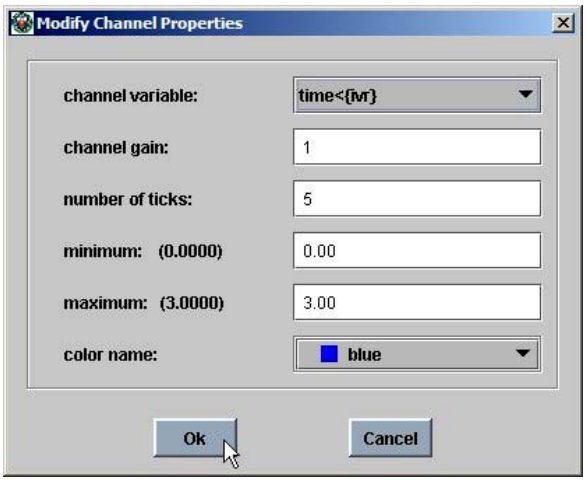

Figure 4.22. Dialog box to create additional graph panes.

This is identical to the time axis for our current display. Click Ok and the new axis will appear on the Output Screen, see Fig. 4.23. Now we need to move the calcium scale to this graph.

Click on Var2<ch1:Cion… in the Variables pane on the right. Once selected, the text is highlighted in blue. Click *Edit>Modify* to open the Modify Variable Properties window. Edit the field Channel number and change the value to 2. Click Ok, and the calcium scale should appear on the second graph on the lower half of the Output screen, as in Fig. 4.23. Click *File>Save*, and rerun the simulation.

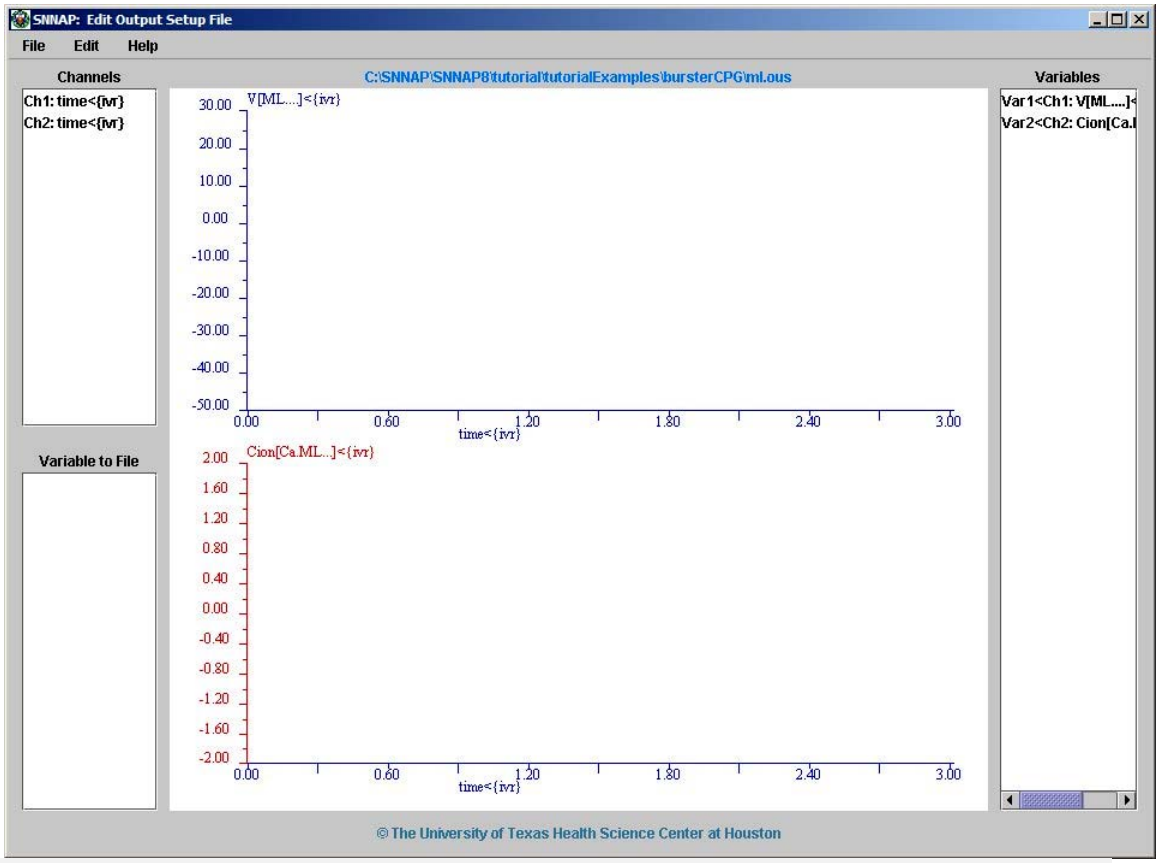

Figure 4.23. Output screen with two graph panes.

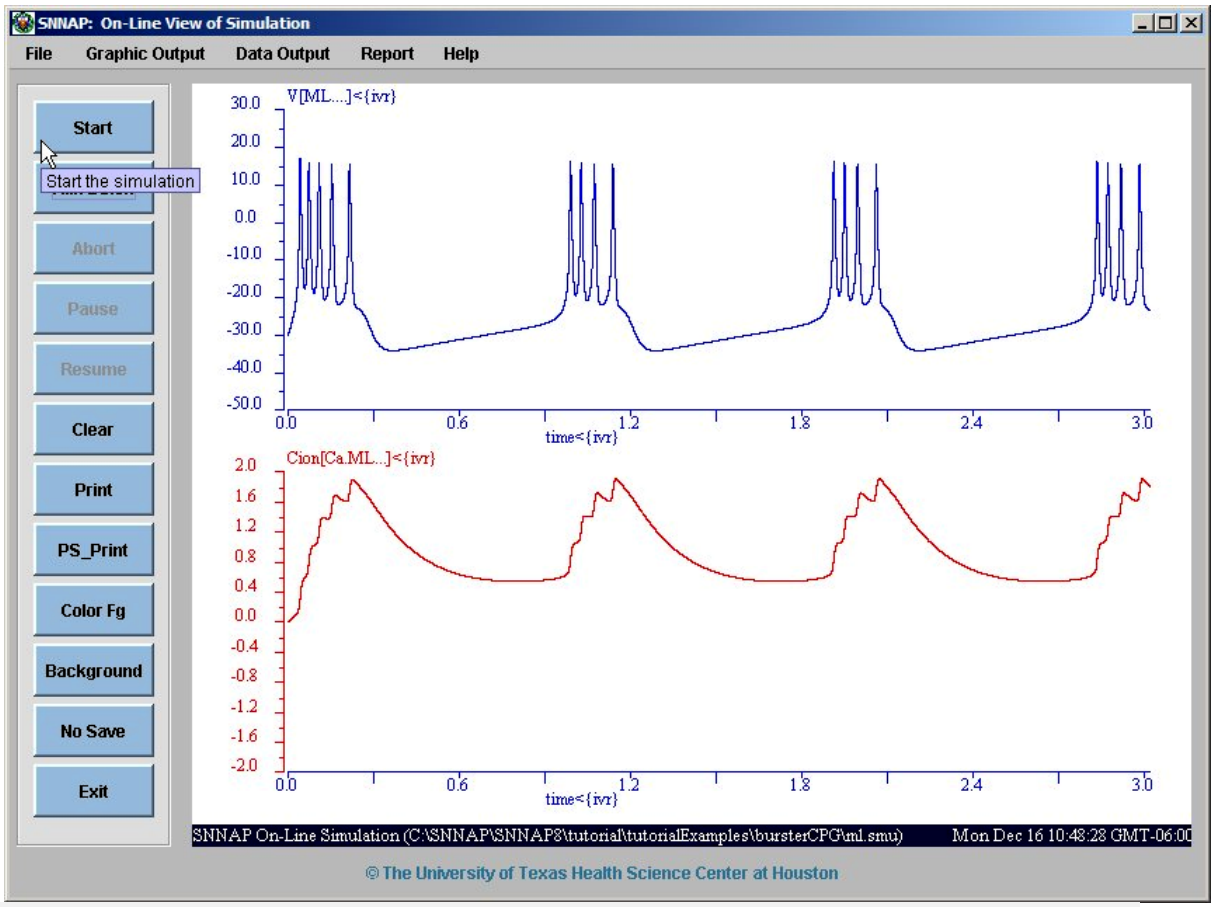

Figure 4.24. Membrane potential and calcium concentration on separate graphs.

Now that we know how to display two separate graphs with SNNAP, lets display the ion currents of our model to better understand the dynamics of the model.

Displaying Ion Currents

Figure 4.24 displays the time course of two model variables, membrane potential and intracellular calcium concentration. Lets add to the display window the major currents of this model, the inward calcium and two outward potassium currents.

As a first step, lets return the calcium concentration plot to the graph with the membrane voltage, and use the second graph for the ionic currents.

Next, we need to declare three new output variable in SNNAP, for the three currents.

Open the Output Screen window and click *Edit>Add>Add Variable* to open the Add One Variable dialog box. Set the parameters as shown in Fig. 4.25 for the three currents.

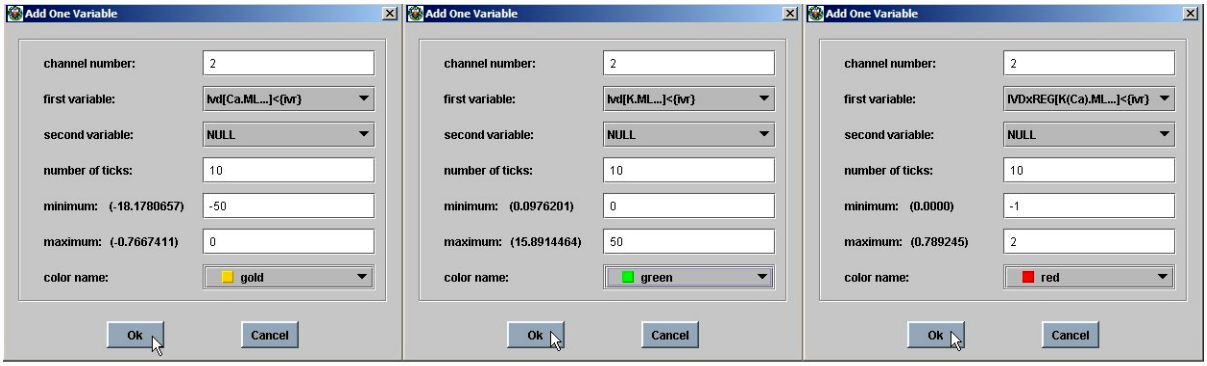

Figure 4.25. To declare three new output variables, three active currents.

Repeat this step three times for variables: *Ivd[Ca.ML]* , *Ivd[K.ML]* and *IVDxREG[K(Ca).ML]*.

The final Output Screen should look like Fig. 4.26. Click File>Save to save the changes in SNNAP.

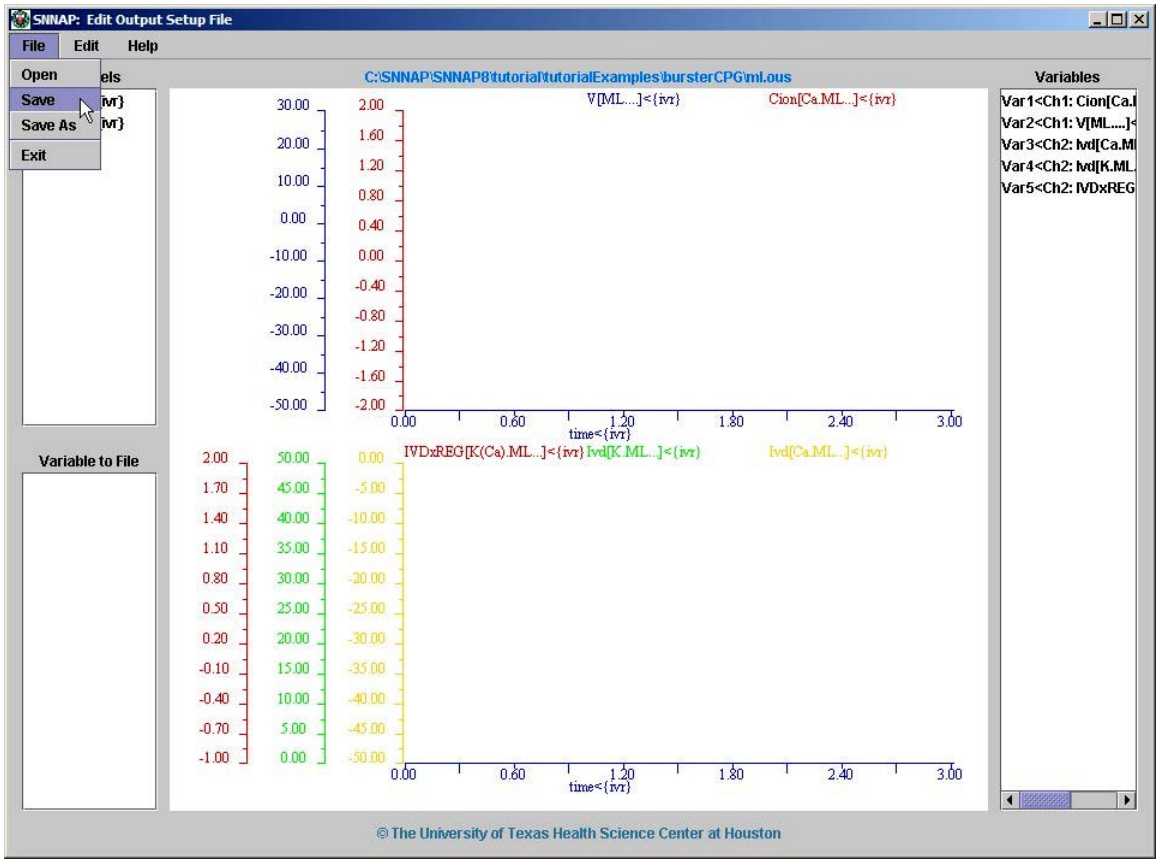

Figure 4.26. Output screen in after addition of current variables for ploting.

Now lets rerun the simulation:

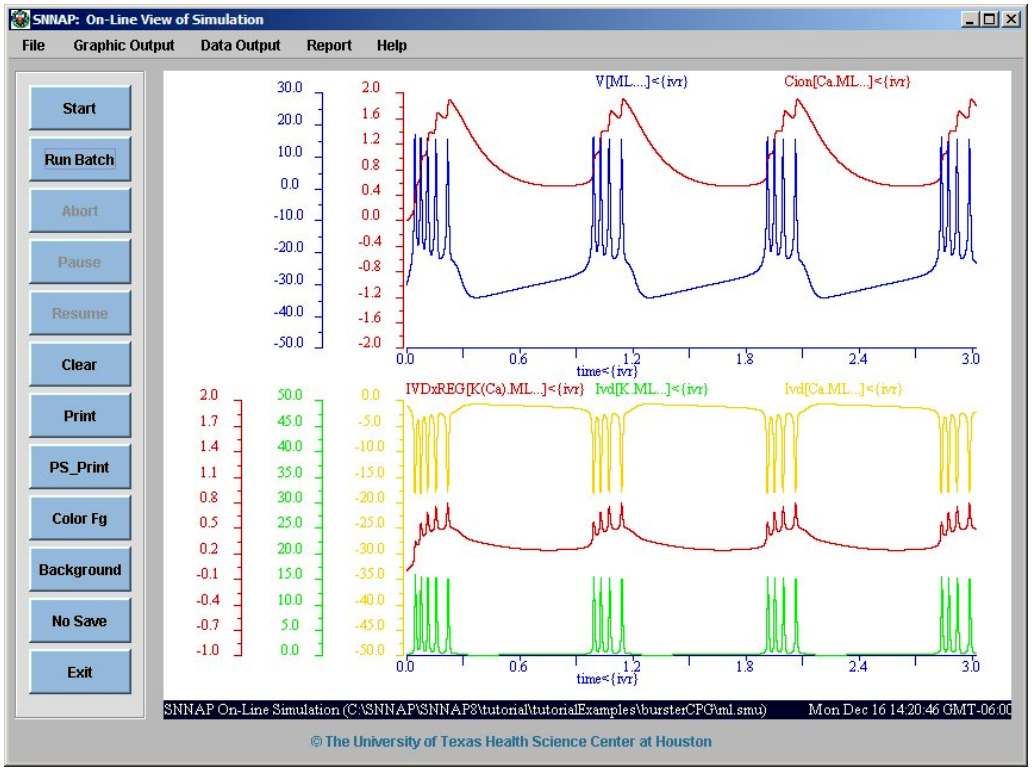

Figure 4.27. Final display of model: two variables and three currents.

CENTRAL PATTERN GENERATOR MODULATION

There are three basic characteristics of a bursting neuron:

- the duration of spiking activity;
- the frequency of action potentials during the burst;
- the duration of the quiescence period.

The period of an entire bursting event is the sum of both active and quiescence durations.

There are various parameters that can alter the behavior of bursting neurons. For example the duration of the burst is dependent on the rate of intracellular calcium

accumulation. By reducing the rate of intracellular calcium accumulation the duration of activity is increased. Figure 4.28 shows a slightly reduced value for parameter k2 of the calcium ion equation (ion formula in Formula Edior window) reduced from 0.4 to 0.35. The burst duration is increased, compare with Fig. 4.27.

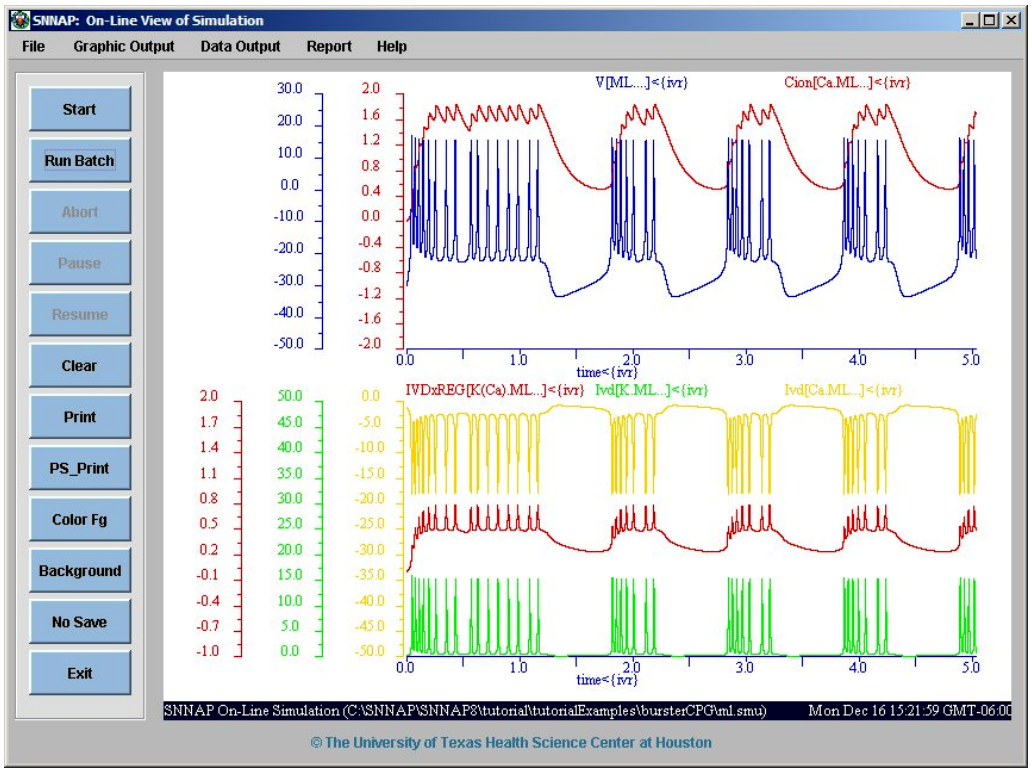

Figure 4.28. Burst modulation by decreasing calcium acc

CHAPTER 5 MODELING SMALL NEURONAL NETWORKS

In this chapter we will learn how to simulate small networks of neurons using SNNAP. A graphical user interface is provided with SNNAP, which allows you to connect neuron models into networks.

The Hodgkin-Huxley model presented in chapter 3 will be used as our neuron model. Our network will be composed of several such neuron models.

Neurons communicate across synapses which may be either excitatory, inhibitory or electric. SNNAP offers you several ways to connect neurons in a network. We will incorporate both excitatory and inhibitory synapses into our network.

This chapter is composed of the following sections:

- *Introduction to Small Neuronal Networks* - A small neural network, composed of three Hodgkin-Huxley neuron models with excitatory connections is organized in a ring architecture. The network may exhibits oscillatory behavior.
- *Synaptic Connections* – Synaptic release is described by an alpha function. By varying the synaptic strength, the network exhibits oscillations of various frequencies. By changing the synaptic connection to be inhibitory, the network can still exhibit oscillatory behavior using a mechanism called *post inhibitory rebound* for action potential generation.
- *Building Networks* - The SNNAP graphical user interface may be used to incorporate neurons and synaptic models in order to construct and enlarge a neural network. We will incorporate an additional Hodgkin-Huxley neuron model to our previous network model and exhibit oscillatory behavior with this four cell network.

INTRODUCTION TO SMALL NEURONAL NETWORKS

In order to understand the basic principles of how neural systems function, we must investigate how neurons interact in a network. In this chapter we will examine a simple network composed of three Hodgkin-Huxley neuron models (see chapter 3). Such neuron models exhibit an action potential as a response to suprathreshold stimulation.

We will simulate a network of three Hodgkin-Huxley (HH) model neurons, connected by excitatory chemical synapse. The network architecture is in the form of a ring. After providing an initial transient stimulus to one neuron, the three neurons fire sequentially, and the network exhibits stable oscillatory behavior.

Running a Small Neuronal Network Simulation

This SNNAP tutorial provides you with a small network you can run and experiment with. Open the main SNNAP window (double-click the file `SNNAP8.jar` in folder SNNAP8) and launch the simulation window (click button `Run Simulation`). Locate the file `hhNetwork.smu` in folder `SNNAPhome\SNNAP8\tutorial\tutorialExamples\hhNetwork`, and run the simulation (see section Running a Simulation).

> Note: For Linux users, you should copy the model in `/SNNAPhome/SNNAP8/tutorial/tutorialExamples/archive/Linux/initialModel/hhNetwork`
>
> to your working directory:
>
> `SNNAPhome/SNNAP8/tutorial/tutorialExamples/hhNetwork`.

Figure 5.1 displays the results of the simulation.

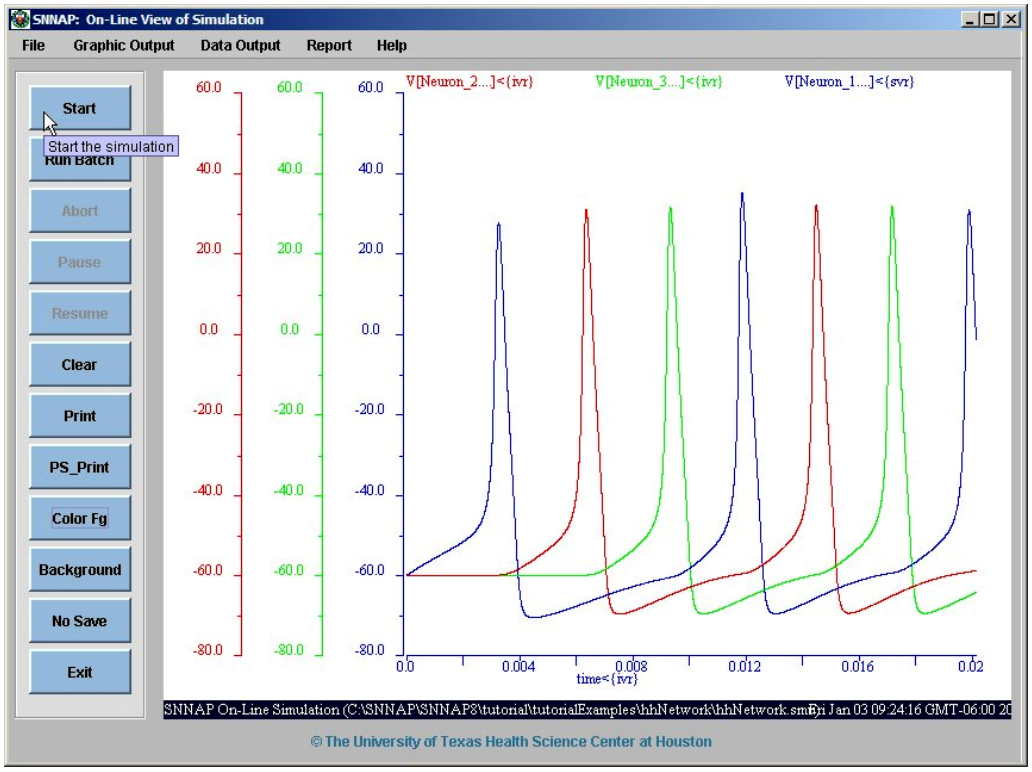

Figure 5.1. Three neuron network connected in a ring architecture.

Figure 5.1 shows the behavior exhibited by the three cell network, while Fig. 5.2 presents the network architecture. The first neuron to fire, blue colored trace, was stimulated by a current injection from time 0 to 0.003 sec. (3msec) with an intensity of 5uA/cm2. This neuron then causes a synaptic current (not shown) which stimulates the second neuron, red colored trace. The neuron shown in red, after firing an action potential, causes a synaptic current which stimulates the third neuron, shown in color green, which fires in turn. Now the third neuron (green) is connected to the first neuron (blue) and thereby stimulates it with a synaptic current, so that the firing cycle begins again.

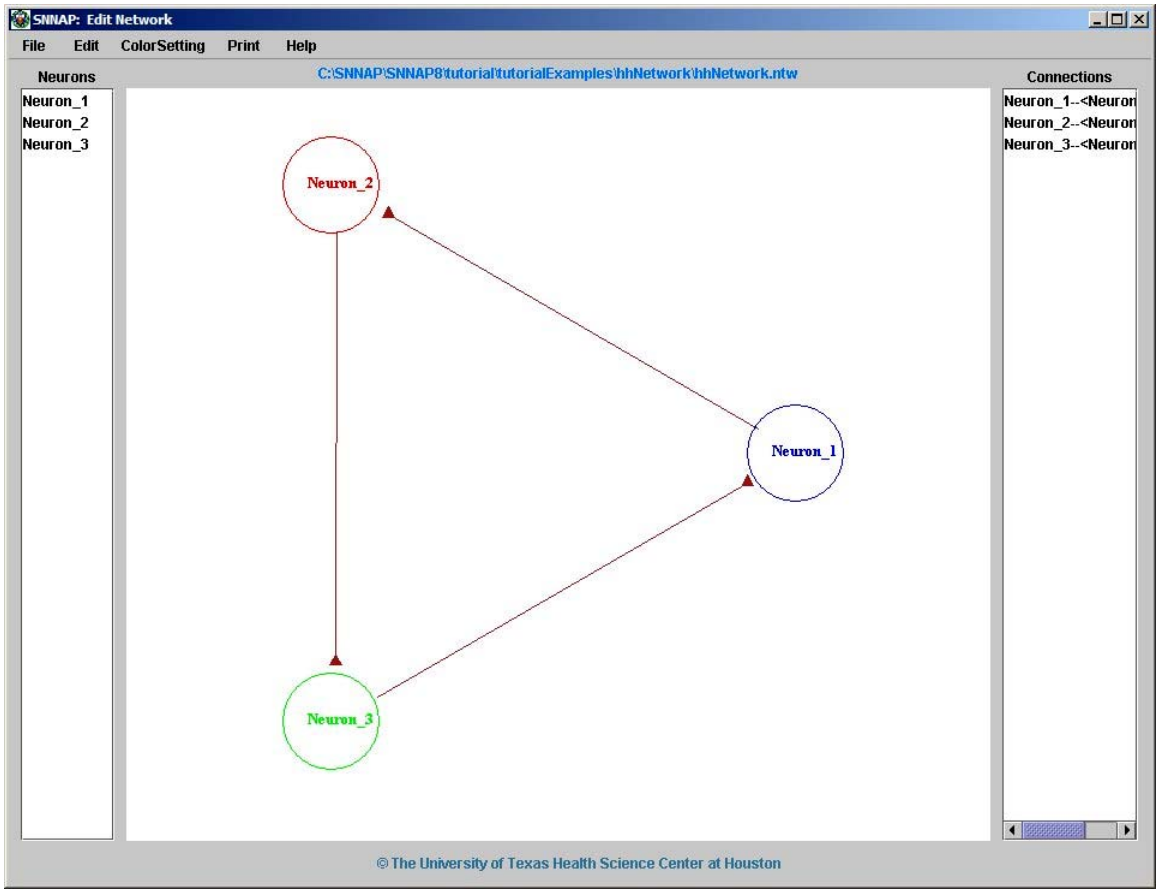

Figure 5.2. Architecture of three neuron network.

The three neuron network is shown in Figure 5.2. The colors of the neurons correspond to the color of the traces in Fig. 5.1. The synaptic connections are marked by the brown lines with triangles at the ends showing the direction of the connection. Therefore we see that Neuron_2 is stimulated by Neuron_1 which in turn is stimulated by Neuron_3, and Neuron_3 receives stimulation from Neuron_2.

SYNAPTIC CONNECTIONS

The three neuron network exhibits oscillatory behavior due to the synaptic connection between the neurons. When a neuron fires an action potential, it causes a synaptic current which stimulates the connected neuron. The synaptic current is modeled by an alpha function multiplied by a maximal conductance and a driving force as follows:

$$I_{syn}(V,t) = \bar{g}_{syn} \cdot g_{syn}(t) \cdot (V(t) - E_{syn}) \qquad (5.1)$$

$$g_{syn}(t) = t \cdot \exp(-t/u) \qquad (5.2)$$

where t is the time that has passed since the "trigger" of the synaptic current, u is the time to peak, E_{syn} is the synaptic reversal potential and the maximal synaptic conductance is $\bar{g}_{syn}$.

In order to display the synaptic current and alpha function, lets add a second graph to the output display. Click the button `Edit OutScreen` (in the main SNNAP window) to edit the output display. Click *File>Open* and select the file `hhNetwork.ous`. Following the instructions in section Changing Output Screen, to add a new channel, with time parameters (min, max) from 0 to 0.02 with 5 tick marks, gain equal 1 and color set to blue. Next add two variables, one for the alpha function, variable fAt (see Fig. 5.3)

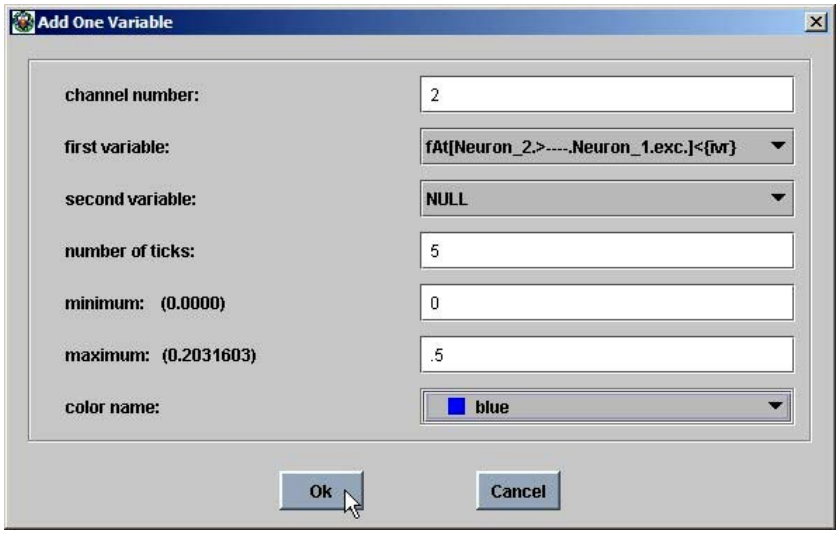

Figure 5.3. Synaptic current variable from neuron 1 to neuron 2.

and a second for the total synaptic current variable Ics (see Fig. 5.4).

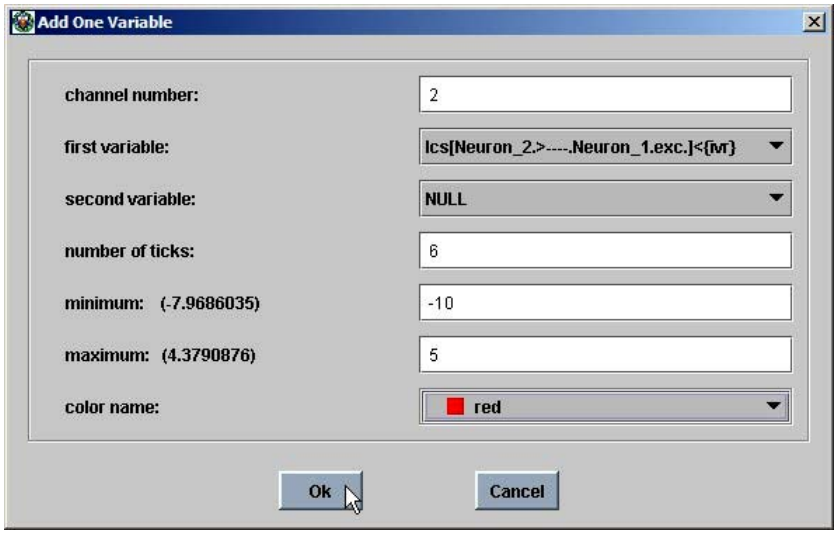

Figure 5.4. Synaptic current in neuron 2.

We can now rerun the simulation and view the trace of the synaptic current and alpha function from neuron one to neuron two.

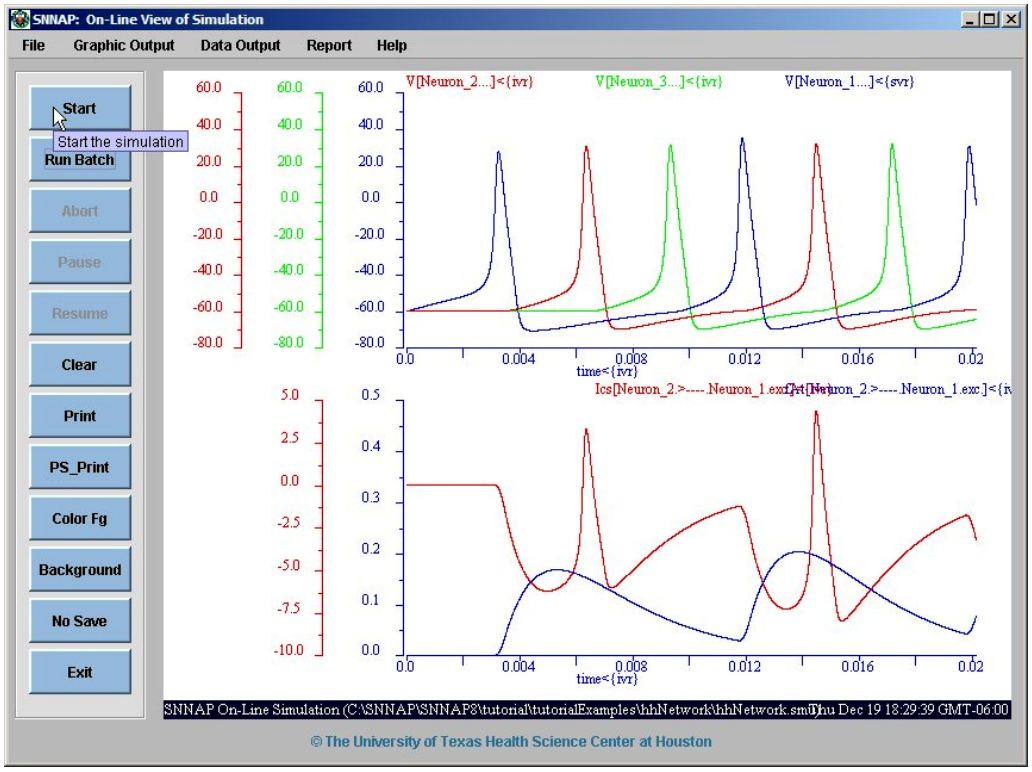

Figure 5.5. Three neuron network with synaptic input and current in neuron 2.

Figure 5.5 presents the synaptic current (red curve in lower trace) from Neuron_1 (blue) to Neuron_2 (red). When the membrane potential of a neuron passes 0mV, it triggers the synaptic potential, and the alpha function (lower blue trace) is calculated, with time *t* increasing from 0, the moment of trigger.

Note - The threshold for triggering the alpha function is a SNNAP parameter which is set in the neuron file, *.neu, edited by clicking the button "Edit Neuron" in the main SNNAP window.

The synaptic current in our case is an excitatory current, which produces an inward current that depolarizes the neuron. The current, having a reversal potential of 0mV has an outward component when the neuron membrane potential is above 0mV.

Similarly, the two other neurons have synaptic connections, as show in Figure 5.2, and follow the same time course as displayed in Fig. 5.5, but with a delay, corresponding to the time of activation.

The synaptic current is represented by two equations in SNNAP, corresponding to equations (5.1) and (5.2).

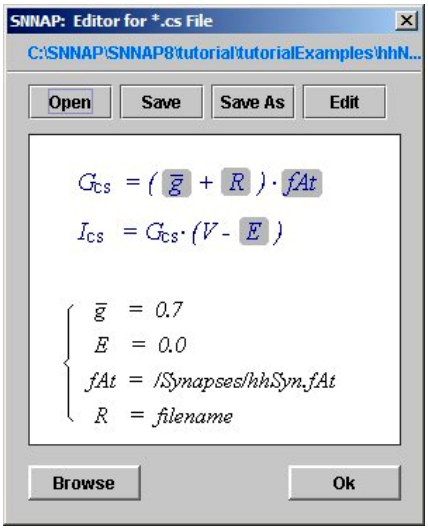

Figure 5.6. Equation for chemical synaptic current.

The synaptic equation is represented by a `*.cs` file. To edit this file, click the `Edit Formula` button on the main SNNAP window, and then click the button `cs` in the formula editor. A file chooser dialog box will open.

Locate the `*.cs` file, in our case, it is
SNNAP8\tutorial\tutorialExamples\hhNetwork\Synapse\hhSyn.cs

The function that is used, the alpha function, is defined in the `*.fAt` file. To edit this file, use the formula editor and click on the button `fAt`. A window as seen in Fig. 5.7 will appear. Clicking on the highlighted parameter `At` will cause the equation window to open as seen on the right panel of Fig. 5.7. The only value to edit with this alpha function is the time to peak, currently set at 0.002 sec (2 msec).

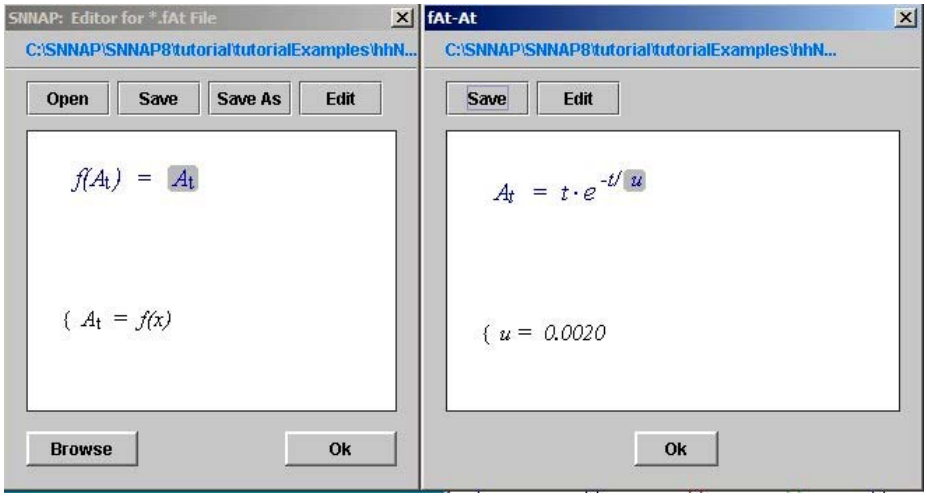

Figure 5.7. Alpha function equation.

There are two parameters that influence the effectiveness of the synaptic current. The parameter u alters the duration of the current while the maximal synaptic conductance $\bar{g}_{syn}$ changes the strength of the current. Lets increase the synaptic strength of the current to 2mS/cm2.

Changing Synaptic Strength

In order to change the chemical synaptic (cs) strength you must open the `*.cs` file through the formula editor. In the main SNNAP window click “Edit Formula”. In the Formula Editor window click “cs”, which will open a file selector window. Select the file *SNNAP8\Tutorial_Examples\HH_Network\Synapse\hhSyn.cs*. Click the parameter $\bar{g}$, which has a grey background. In the parameter window which opens, change the value of $\bar{g}$ to 2 (mS/cm2). Click `Ok` in the parameter window, and in the cs editor window (you will be asked to confirm the changes made, click `Yes`).

Now rerun the simulation by clicking *File>Reload Simulation* in the Simulation window.

Click the `Start` button. You will see that the synaptic current has a range larger than the range we provided for in the output screen. Lets increase that range to be (min,max) = (-35,15).

In the main SNNAP window, click the `Edit OutScreen` button to open the `Edit Output File` window. In the output window, Click *File>Open* and select the file `hhNetwork.ous` in the folder *SNNAP8\tutorial\tutorialExampes\hhNetwork.* The output window should look like Fig. 5.8. Click in the `Variables` pane, the entry `Var5<Ch2:Ics…` to select the synaptic output variable. Click *Edit>Modify* to open the `Modify Variable Properties` dialog box. Set the `minimum` and `maximum` values to `-35` and `15` respectively. Click `Ok` to close the dialog box. Click *File>Save* in the output window.

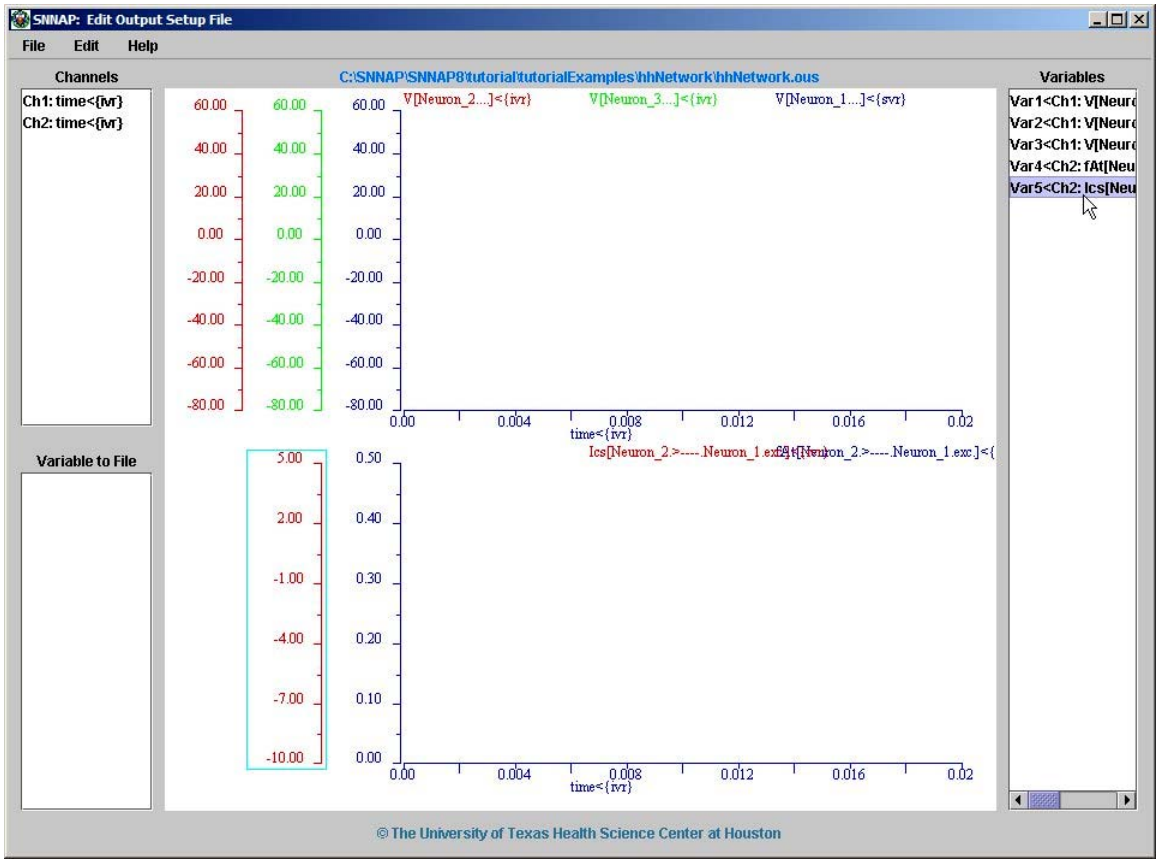

Figure 5.8. Output screen window with selected variable.

Now rerun the simulation by clicking *File>Reload Simulation* in the Simulation window.

Click the `Start` button in the simulation window. Figure 5.9 shows the display you should see on your screen.

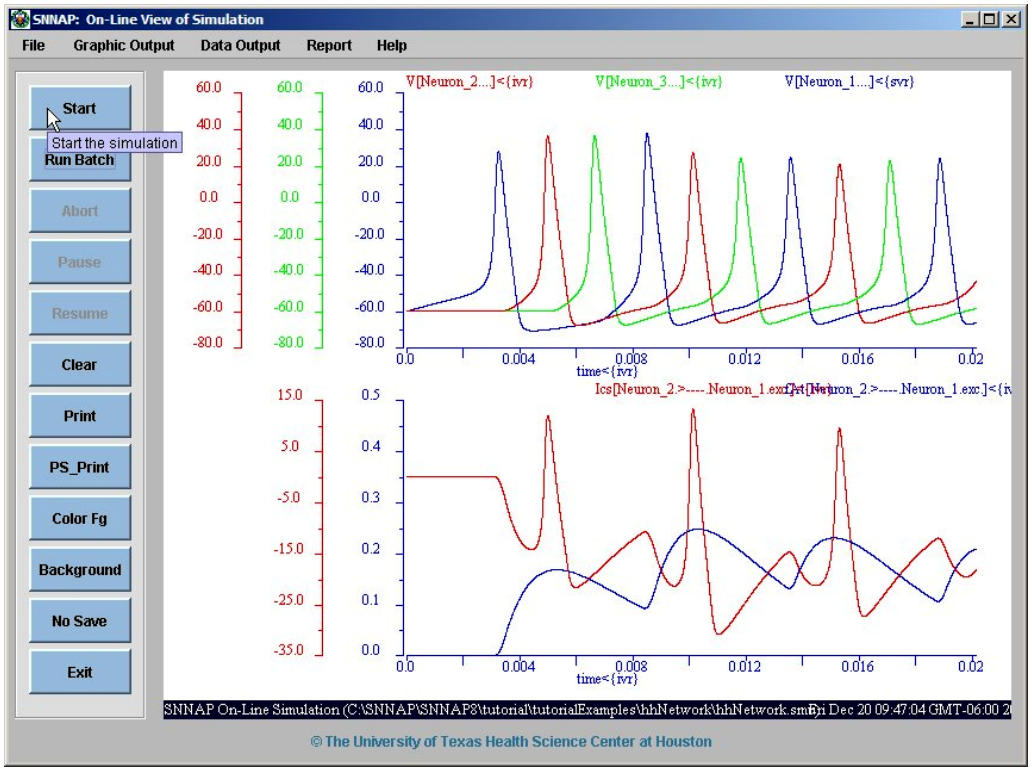

Figure 5.9. Three neuron network with synaptic conductance of 2mS/cm^2.

Notice that the network oscillates at a higher frequency than previously. The neurons increased their firing frequency from 125Hz (Fig. 5.5) to 167Hz. Lets change the maximal synaptic conductance to 5mS/cm2.

Repeat the process described above, and set the $\bar{g}$ value to 5 (mS/cm2).

Rerun the simulation.

(You should also changed the synaptic output variable to a (min,max) = (-80,40) range.)

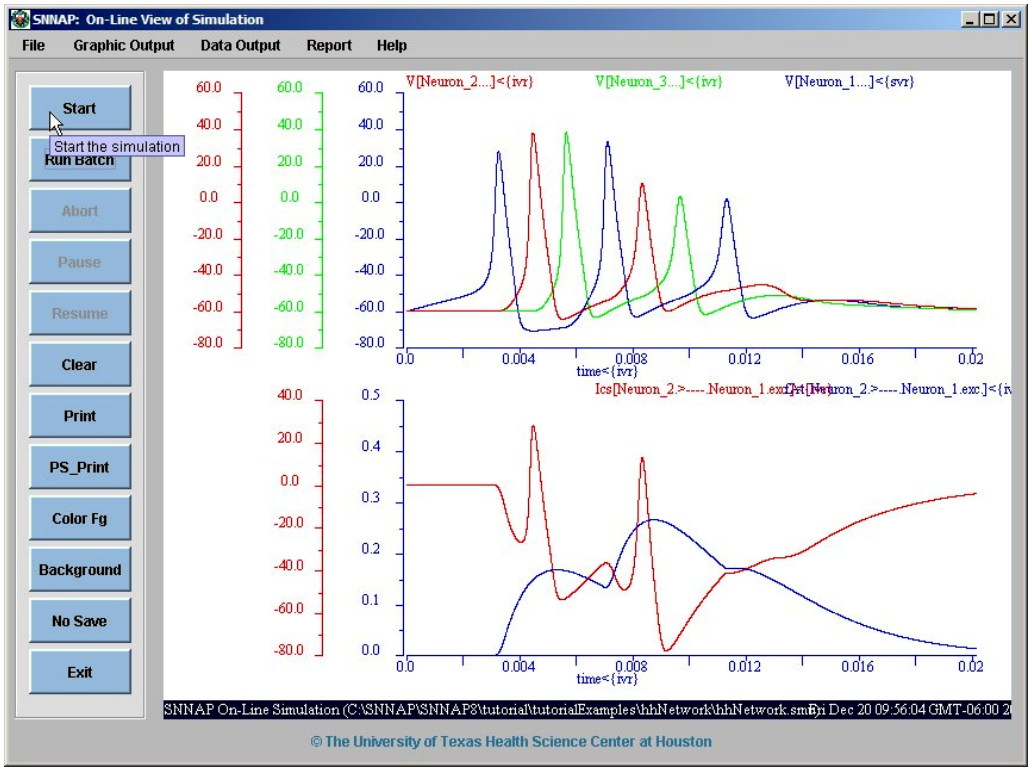

Figure 5.10. Three neuron network with synaptic conductance of 5mS/cm^2.

What has happened? The increased synaptic conductance causes the oscillations to extinguish. For the blue neuron, the first interspike interval is 4ms (250Hz) while the second interspike interval is even shorter, because of the increase in synaptic current (see red curve in lower trace). Finally, the interspike interval became shorter than the refractory period of the neuron and therefore the neurons stopped firing.

Exercise – Using a time to peak time of 1msec, what is the highest oscillating frequency possible for this network.

BUILDING NETWORKS

SNNAP offers a graphical user interface to help you construct a network. Constructing a network is essentially a two step process. The first step is to declare neuron models

which will represent the neurons of the network. The same model may be used for several neurons or various models may be used. The second step is to connect the neurons into a network. The links may be one of three types of synaptic connections. The equations governing the type of connections are provided in the equation library (`formulaTemplates` folder).

Adding a Neuron to a Network

We will now add a fourth neuron to our three neuron network.

Adding a neuron to a network, from a library of neurons that already exist, only requires us to modify the network, through the network editor. Open the network editor by clicking the button `Edit Network` in the main SNNAP window. Click *File>Open* to open the file chooser dialog box. Click on the file name `hhNetwork.ntw` in folder *SNNAP8\tutorial\tutorialExamples\hhNetwork* and click the button `Open`. Click the menu *Edit>Add Neuron>Add H&H Neuron* as shown in Fig. 5.11.

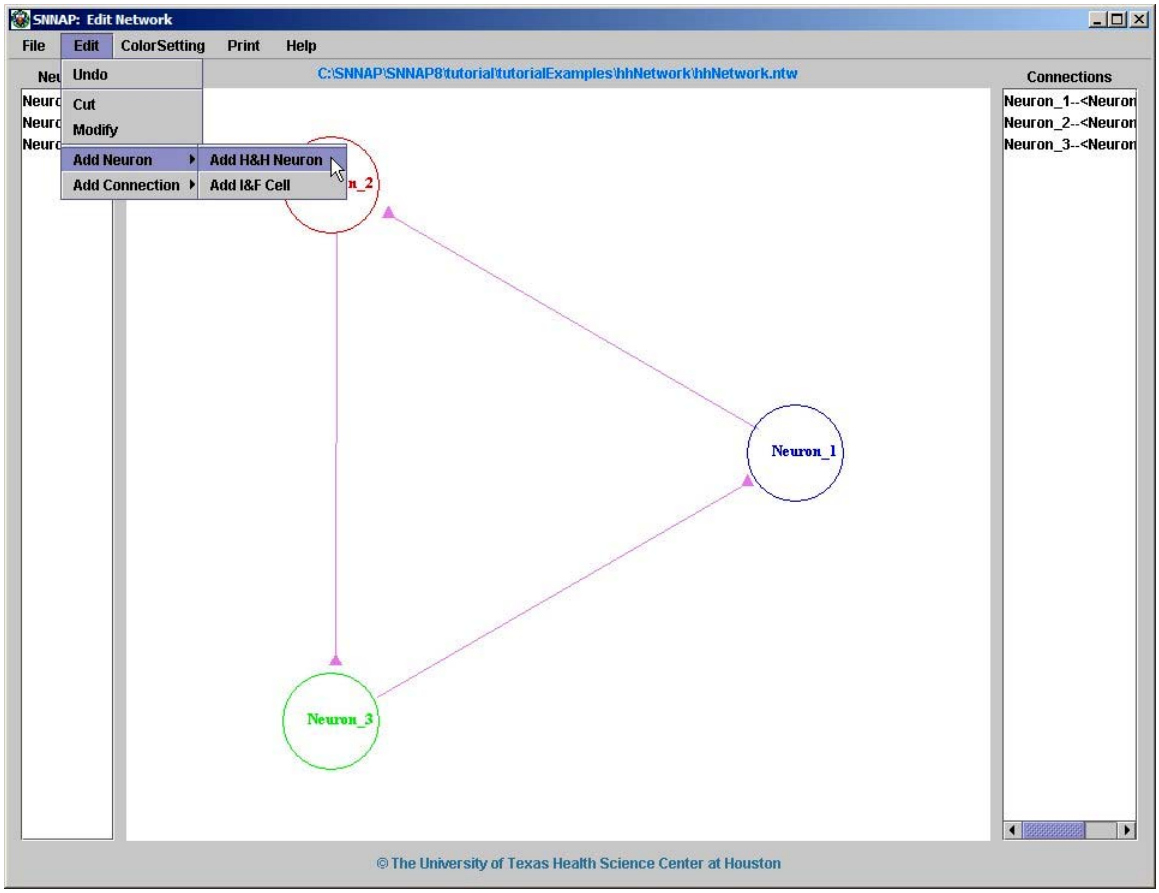

Figure 5.11. In the Edit Network window, add a neuron.

The `Add H&H Neuron` dialog box will open. Enter the values as seen in Fig. 5.12, you may choose any cell name you prefer, as well as color. The important parameter is the file for the neuron description, which should be *Neurons/hhNeuron.neu*. Note that we must provide the path from the folder where the **.smu* file resides, using Unix-like slash ("/") rather than Windows-like back slash ("\").

> Note: For Linux users, you must provide the pathname from the folder where `snnap8.jar` is located, i.e. `tutorial/tutorialExamples/hhNetwork/Neurons/hhNeuron.neu`

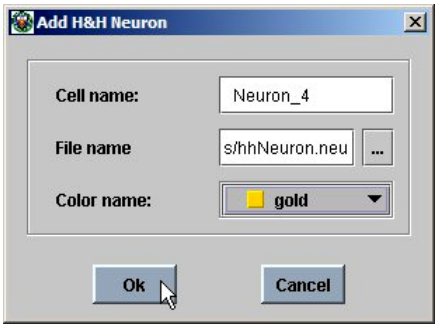

Figure 5.12. Dialog box to enter neuron name and respective equation file.

Once we are satisfied with the parameter values, click the button `Ok`.

The fourth neuron will now appear as part of the network, as shown in Fig. 5.13.

Adding a Synaptic Connection in a Network

We now must connect this neuron to the other neurons of the network. In the `Edit Network` window, click the menu *Edit>Add Connection>Add Chemical*, as seen in Fig. 5.14, to open the dialog box `Add Chemical`.

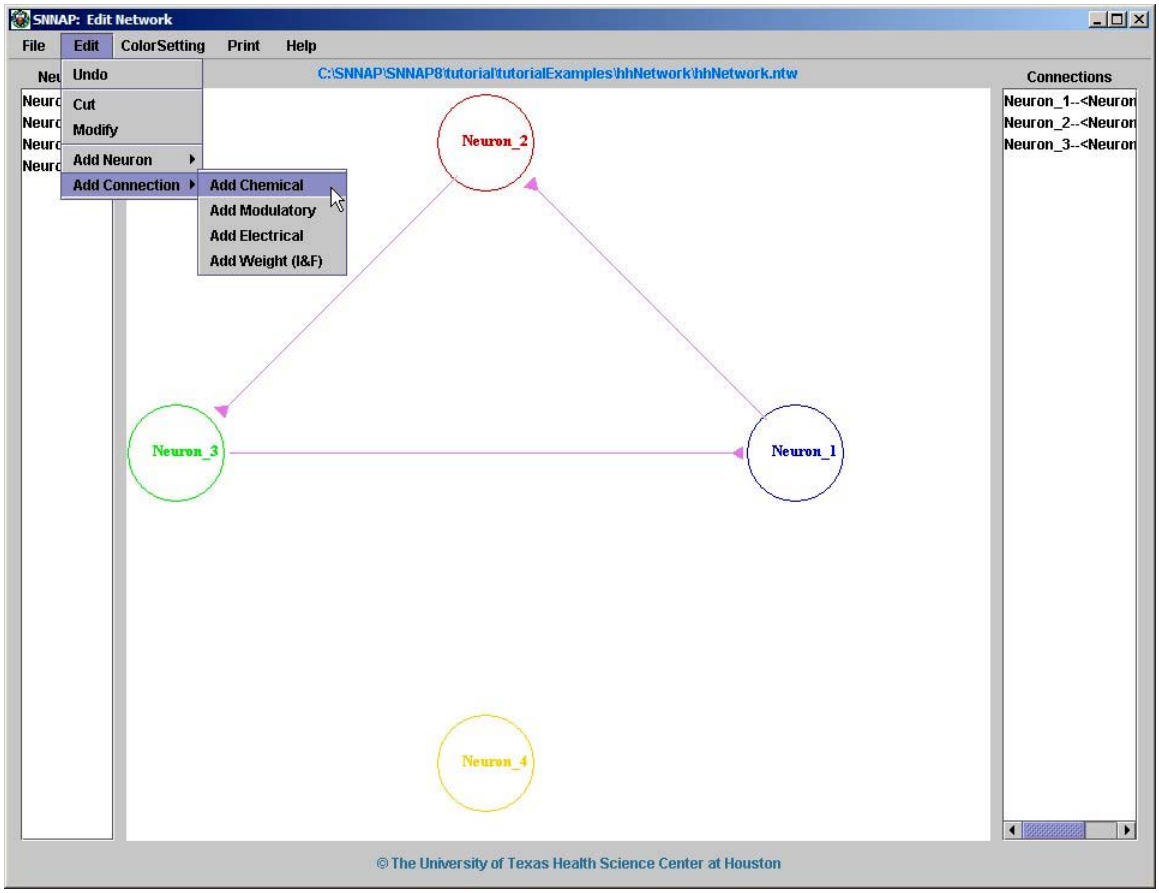

Figure 5.14. Add a synaptic connection from Neuron_4 to the network.

In the *Add Chemical* window, we define the synaptic connection. There are two connections that need to be defined, from Neuron_3 to Neuron_4 (see Fig. 5.15) and from Neuron_4 to Neuron_1. For both connections we will use the same synaptic model file, which is *Synapses/hhSyn.cs*. The parameter `Receptor type` is a label you may use to distinguish various synaptic connections, which appears in the pane `Connections` of the Network window. Also, the connection from Neuron_3 to Neuron_1 should be deleted. Select this connection in the `Connections` pane of the `Edit Network` window and click *Edit>Cut*.

Note: For Linux users, you must provide the pathname from the folder where `snnap8.jar` is located, i.e. `tutorial/tutorialExamples/hhNetwork/Synapses/hhSyn.cs`

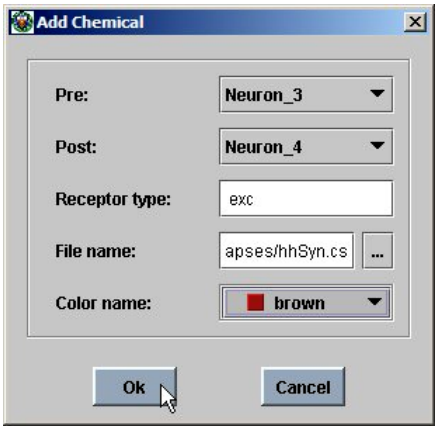

Figure 5.15. Chemical synapse dialog box.

After providing the parameters for the two connections and deleting the connection (Neurons 3->1), the network window should look like Fig. 5.16. Click the menu item *File>Save* to save the network changes in SNNAP.

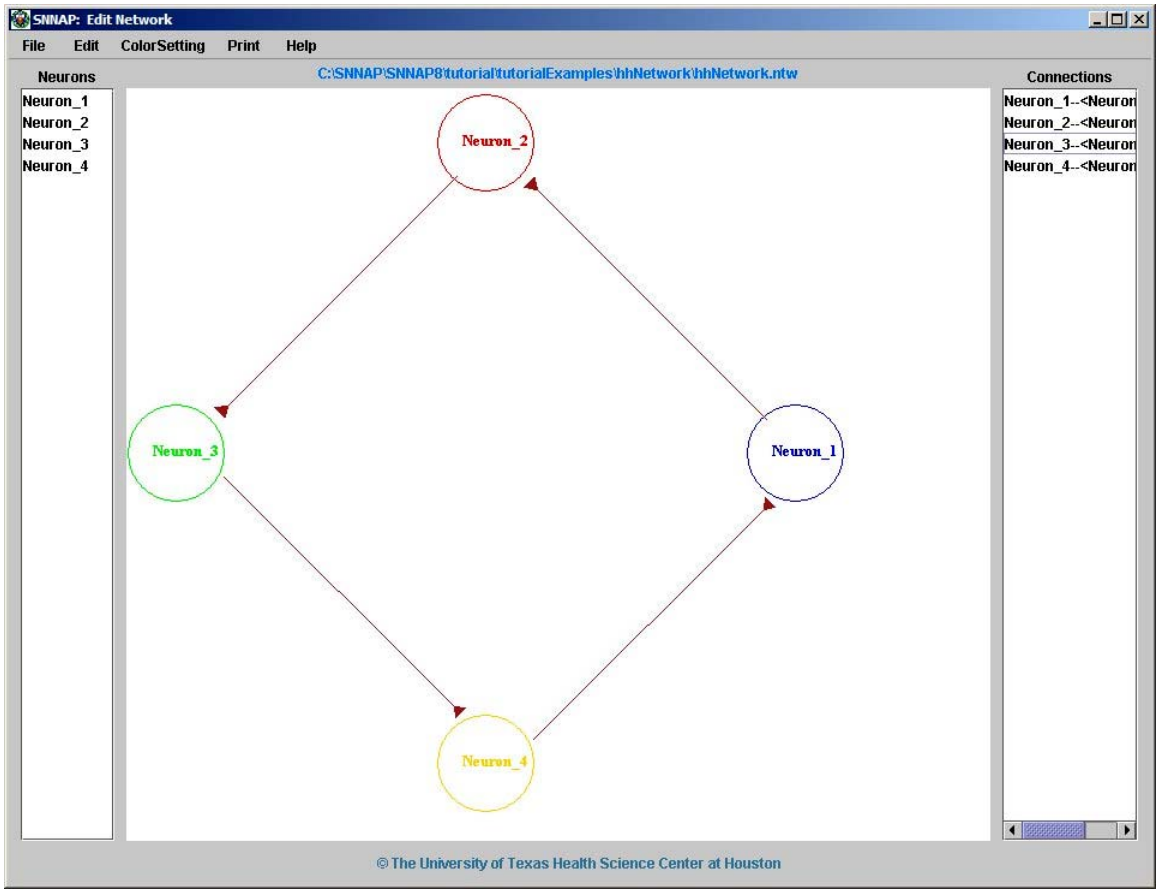

Figure 5.16. Four neuron network.

The last step is to add the fourth neuron to the variable list of the output screen. Following the procedure in section Adding New Output Variable, Chapter 4. Add a new variable (V[Neuron_4…]) to the channel 1 trace, with `number of tick marks` set to 7, and (min,max) values equal to (-80,60). (Set the color to gold, or any color you desire.) Confirm the changes by clicking `Ok` and confirming `Yes`, and now rerun the simulation.

Figure 5.17 shows the behavior of our four neuron network. The firing frequency of the neurons are less than 125Hz (see Fig. 5.5) because of the added neuron in the ring.

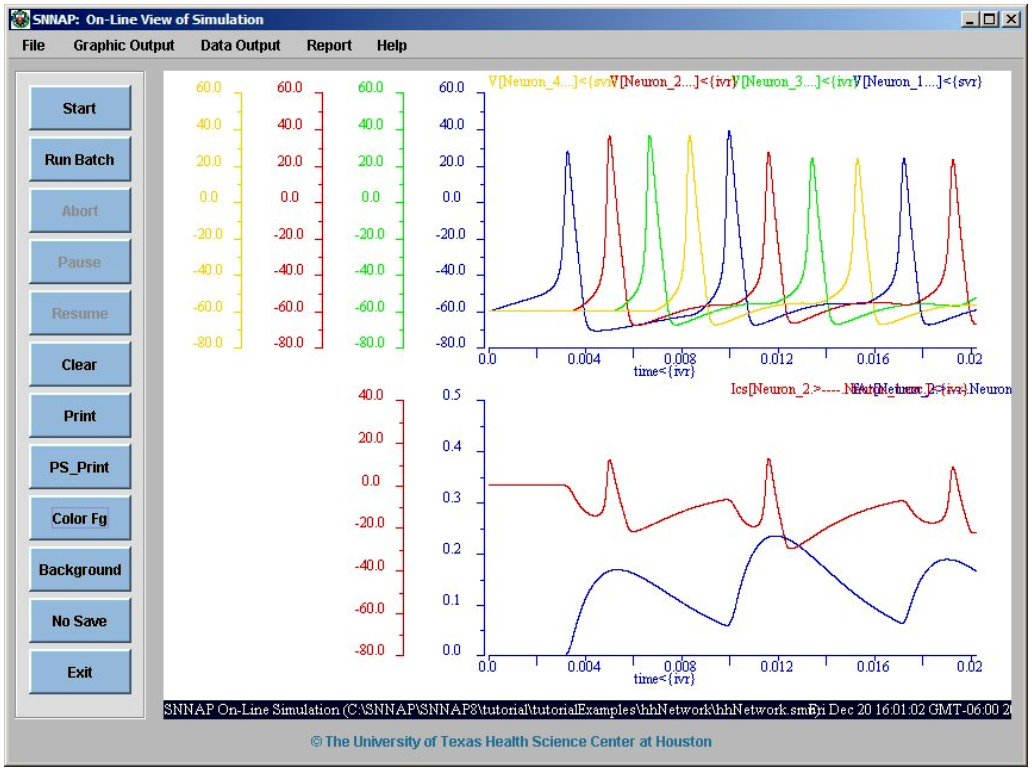

Figure 5.17. Four neuron network with synaptic conductance of 2mS/cm^2.

CONCLUSIONS

We have seen how to construct a network with SNNAP.

The graphical user interface is intuitive and easy to use.

The connections between neurons may be chosen from a library of equations and specific parameters provided by the user.

APPENDIX: REFERENCES

PUBLISHED STUDIES THAT USED SNNAP

1. Baxter, D.A., Canavier, C.C. and Byrne, J.H. (2002) Dynamical properties of excitable membranes. In: Byrne, J.H. and Roberts, J. (Eds.), *Cellular and Molecular Neuroscience* (Chapter 7). New York: Academic Press.

2. Baxter, D.A., Canavier, C.C., Clark, J.W. and Byrne, J.H. (1999) Computational model of the serotonergic modulation of sensory neurons in *Aplysia*. J. Neurophysiol., 82: 2914 - 2935.

3. Baxter, D.A., Susswein, A.J. and Byrne, J.H. (2002) Mechanisms of pattern generations underlying fictive feeding in Aplysia. J. Neurophysiol. (under review).

4. Byrne, J.H., Cleary, L.J. and Baxter, D.A. (1990) Aspects of the neural and molecular mechanisms of short-term sensitization in *Aplysia*: modulatory effects of serotonin and cAMP on duration of action potentials, excitability and membrane currents in tail sensory neurons. In: L. Squire and E. Lindenlaub (Eds.), *The Biology of Memory* (pp. 7-28). New York: F. K. Schattauer Verlag.

5. Cai, Y., Baxter, D.A. and Crow, T. (2002) Computational study of enhanced excitability in *Hermissenda* type-B photoreceptors underlying one-trial conditioning: Role of conductances modulated by serotonin. Biophy. J. (under review).

6. Cataldo, E., Byrne, J.H., Baxter, D.A. and Brunelli, M. (2002) Conduction block at axonal branch points in touch mechanoafferents (T cells) of the leech. J. Neurophysiol. (In preparation).

7. Flynn, M.C., Cai, Y., Baxter, D.A. and Crow, T. (2002) Computational study of *Hermissendia* type-B photoreceptor: Role of action potential duration in synaptic facilitation. Biophy. J. (under review).

8. Kabotyanski, E.A., Ziv, I., Baxter, D.A. and Byrne, J.H. (1994) Experimental and computational analyses of a central pattern generator underlying aspects of feeding behavior in *Aplysia*. Netherlands J. Zool., 44: 357-373.

9. Komendantov, A.O. and Kononenko, N.I. (2000) Caffeine-induced oscillations of the membrane potential in *Aplysia* neurons. Neurophysiology, 32: 77-84.

10. Pelz, C., Jander, J., Rosenboom, H., Hammer, M. and Menzel, R. (1999) I-A in Kenyon cells of the mushroom body of honeybees resembles shaker currents: Kinetics, modulation by K^+, and simulation. J. Neurophysiol. 81: 1749-1759.

11. Steffen, M., Amini, B., Feigenspan, A., Seay, C., Feigenspan, A., Cai, Y., Baxter, D.A. and Marshak, D. (2002) Spontaneous activity of dopaminergic neurons in the mouse retina: a computer model. Biophys. J. (under review).

12. Susswein, A.J., Hurwitz, I., Thorne, R., Byrne, J.H. and Baxter, D.A. (2002) Mechanisms of pattern generation underlying fictive feeding in *Aplysia*. The initiation and maintenance of protraction via coupling between a large neuron with only plateau-like activity and a small conventional neuron. J. Neurophysiol., in press.

13. White, J.A., Ziv, I., Cleary, L.J., Baxter, D.A. and Byrne, J.H. (1993) The role of interneurons in controlling the tail-withdrawal reflex in *Aplysia*: a network model. J. Neurophysiol. 70: 1777-1786.

14. White, J.A., Baxter, D.A. and Byrne, J.H. (1994) Analysis of the modulation by serotonin of the voltage-dependent potassium current in sensory neurons of *Aplysia*. Biophys. J. 66: 710-718.

15. Ziv, I, Baxter, D.A. and Byrne, J.H. (1994) Simulator for neural networks and action potentials: description and application. J. Neurophysiol. 71: 294-308.

LITERATURE CITED IN MANUAL

Hayes RD, Byrne JH, Baxter DA (2003) Neurosimulation: Tools and Resources in The Handbook of Brain Theory and Neural Networks 2nd ed. Arbib MA, MIT Press, Cambridge Massachusetts.

Hodgkin AL, Huxley AF (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. J Physiol 117:500-544

Morris C, Lecar H (1981) Voltage oscillations in the barnacle giant muscle fiber. Biophys J 35:193-213

Rinzel J, Ermentrout B (1998) Analysis of neural excitability and oscillations, in *Methods in Neuronal Modeling: From Ions to Networks*, Koch C, Segev, I (Eds.) 2nd edition, Cambridge: The MIT Press.